LEVEL II

AD A109134

DTIC
ELECTE
DEC 31 1981
S D
H

DTIC FILE COPY

Center for Information Systems Research
Massachusetts Institute of Technology
Alfred P. Sloan School of Management
50 Memorial Drive
Cambridge, Massachusetts, 02139
617 253-1000

8112 31103

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER Technical Report #6	2. GOVT ACCESSION NO. AD-A109 134	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A Study of the Multicache-Consistency Problem in Multi-Processor Computer Systems		5. TYPE OF REPORT & PERIOD COVERED 6. PERFORMING ORG. REPORT NUMBER M010-8109-06 TR-6
7. AUTHOR(s) Tarek K. Abdel-Hamid Stuart E. Madnick		8. CONTRACT OR GRANT NUMBER(s) N00039-78-G-0160
9. PERFORMING ORGANIZATION NAME AND ADDRESS Sloan School of Management Massachusetts Institute of Technology Cambridge, Massachusetts 02139		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE September 1981 13. NUMBER OF PAGES 140
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Bus, Cache, Multicache-consistency, Multiprocessors, Performance Evaluation, Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This paper is a report on an ongoing research project at the M.I.T. Sloan School of Management to study the multicache-consistency problem in multi-processor computer systems.

The nature of the consistency problem in multicache memory systems is briefly discussed, together with an explanation of the three common approaches proposed in the literature to handle it. A new solution to the problem, called the "Common-Cache / Pended Transaction Bus" (CC/PTB)

continued on reverse side

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-014-6601

381699

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. Abstract

approach, is developed and discussed. The "CC/PTB" approach attempts to minimize performance degradation by eliminating the overhead of maintaining cache-consistency. Its two distinctive features are: Firstly, the conventional private cache per processor organization is replaced by one where a pool of cache-modules is commonly shared by all processors. Secondly, the Pended Transaction Bus (PTB) is used as the interconnection protocol that connects the processors and the cache-modules.

The performance of the CC/PTB approach is evaluated using a highly detailed simulation model with favorable results.

Accession For
NTIS GRA&I
DTIC TAB
Unannounced
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A

ABSTRACT

This paper is a report on an ongoing research project at the M.I.T. Sloan School of Management to study the multicache-consistency problem in multi-processor computer systems.

The nature of the consistency problem in multicache memory systems is briefly discussed, together with an explanation of the three common approaches proposed in the literature to handle it. A new solution to the problem, called the "Common-Cache / Pended Transaction Bus" (CC/PTB) approach, is developed and discussed. The "CC/PTB" approach attempts to minimize performance degradation by eliminating the overhead of maintaining cache-consistency. Its two distinctive features are: Firstly, the conventional private cache per processor organization is replaced by one where a pool of cache-modules is commonly shared by all processors. Secondly, the Pended Transaction Bus (PTB) is used as the interconnection protocol that connects the processors and the cache-modules.

The performance of the CC/PTB approach is evaluated using a highly detailed simulation model with favorable results.

TABLE OF CONTENTS

I. THE MULTICACHE-CONSISTENCY PROBLEM: AN INTRODUCTION

A "consistency problem" refers, in general, to a situation where two or more entries representing the same "fact" in a data base differ (i.e., are inconsistent). This, of course, can only occur when redundancy exists. In this paper we will be concerned with the "consistency problem" that arises in cache-based memory systems.

A cache memory system (Kaplan and Winder, 1973) represents a type of memory hierarchy that ttempts to bridge the CPU-main memory speed gap by the use of a small, high speed random access memory whose cost per bit is higher than that of main memory, but whose total cost is relatively small because of the small size. Conceptually, this configuration has analogies with paging systems (Matick, 1977). The implementations, however, are far apart because of speed considerations. In contrast to a paging system, a cache is managed by hardware algorithms, provides a smaller ratio of memory access times (e.g., 10:1 rather than 1000:1), and deals with smaller

blocks of data (64 bytes for example rather than 4096).

In a cache system, all data are referenced by their main memory address. At any given time, a certain subset of the contents of main memory is contained in the cache level. If a processor then requests a data item in this subset, the request is serviced at the cache level.

A cache-based system works "well" for two basic reasons: First, executing programs tend to re-use instructions and data; and second, programs tend to use instructions and data near recently used instructions and data. The first property means that once information is fetched from main memory to cache, subsequent accesses to it are at cache speed. The second property means that if a request to main memory is satisfied by bringing into a cache a block of information larger than is immediately needed, the additional information is likely to be needed soon, and its presence in the cache will save one or more references to main memory.

In this paper we will refer to the block of information that constitutes the minimum amount of data which may be transmitted between the cache and main memory and which is also the allocation unit in the cache as the "cache line." All bytes of a cache line are, therefore, simultaneously all present or all absent from the cache. A directory is usually used to record the main memory addresses of all lines in the cache.

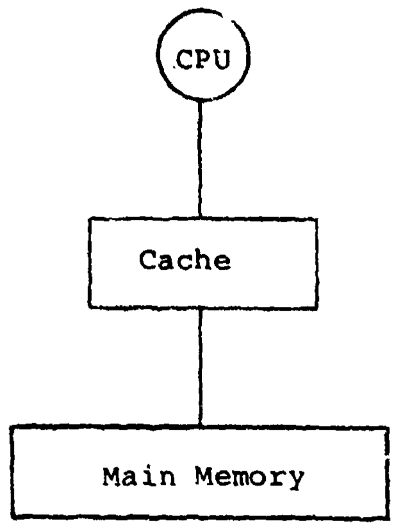
It is easy to demonstrate how inconsistency can develop in

cache-based systems due to the existence of redundancy. Consider first the simple case of a single-cache organization (Figure 1(a)).

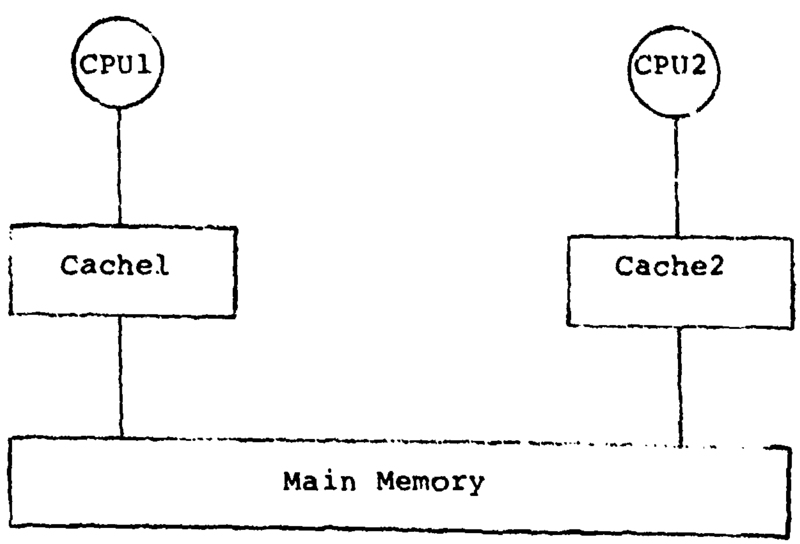
The CPU can only access words that are in the cache. If a word that is needed for processing is not already in the cache, it will first have to be transferred from main memory to the cache. Once the word is in the cache it becomes accessable by the CPU and processing can take place. If the CPU then updates (i.e., modifies) the word, inconsistency between the copy in the cache and the copy in main memory could develop. This depends on the store algorithm used.

If a store through algorithm (in which the cache and main memory are updated simultaneously) is used, inconsistency will not arise. The price paid for that is a decrease in processing speed as store operations become limited by the speed of main memory. When this price is too high, store-behind or store-replacement algorithms may be used. In both cases main memory is not updated immediately, and as a result inconsistency arises. The modified word in the cache will, for "some" interval of time, be different from its unmodified version in main memory.

The more interesting case, however, is that of multicache systems. Consider the two-cache organization of Figure 1(b). What is important to emphasize here, is that the store-through algorithm is no longer sufficient to avoid inconsistency. Assume, for example, a word whose main memory address is (A), and whose current value is (V) is present in both cache1 and cache2. CPU1 then



(a) Single-Cache Organization



(b) Two-Cache Organization

Figure (1)

modifies the value of the word in its cache1 to (V.), and assuming a store-through algorithm is used, main memory is simultaneously updated. However, cache2 continues to have the unmodified version (V). If, before this inconsistency is resolved by either updating, replacing, or invalidating cache2.s copy, CPU2 attempts to access the word (A), it will get what is now an invalid value (V) from its cache2.

It is time now to adopt what we believe is a more useful definition of what a "consistency problem" is. We claim that inconsistency per se is not necessarily a problem. Reconsider the case of a single-cache organization. We have already explained how inconsistency can arise for "some" interval of time when the store through algorithm is not used. During that interval of time the cache will contain the modified version of the word, while main memory will not. If, during this interval, the CPU needs to re-access the word for processing, what will happen? It will check the cache, find the word in it, and, therefore, access it i.e., no transfer from main memory will be needed. Thus, although inconsistency exists, no problem arises because the CPU will always access the updated version of the word from the cache.

With this in mind, we now adopt the following definition of a "consistency problem" (Censier and Feautrier, 1978):

> "A consistency problem exists in a cache-based system if the value accessed by a CPU is not the value given by the latest store operation (by any CPU) to the same address."

It is obvious, from the above discussion, that there will be no consistency problem for single-cache organizations. These, unfortunately, are not very attractive for high performance systems.

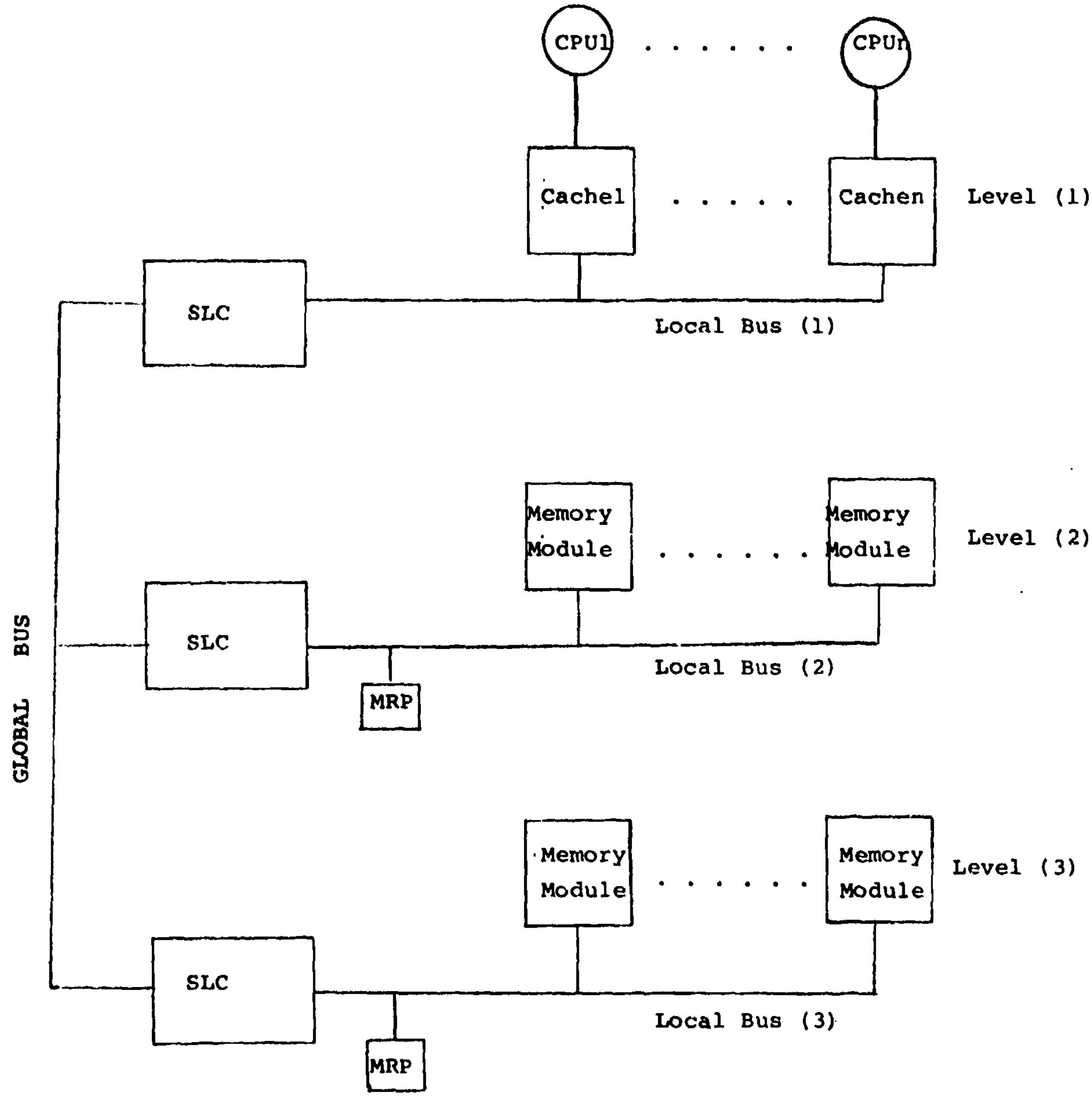
In the next part of this paper we present a brief discussion of INFOPLEX, a highly parallel multi-processor computer system that utilizes a multicache organization. In such an organization the consistency problem is a significant one. The INFOPLEX organization will constitute the context within which we shall evaluate the different approaches for handling the multicache-consistency problem.

II. INFOPLEX: A MULTI-PROCESSOR COMPUTER SYSTEM

A research project is currently underway at the MIT Sloan School of Management aimed at investigating the architecture of a new data base computer, called INFOPLEX, which is particularly suitable for large-scale information management (Madnick, 1979). The specific objectives of the project include providing substantial performance improvements over conventional architectures, supporting very large complex data bases, and providing extremely high reliability.

To provide a high performance, highly reliable, and large capacity storage system, INFOPLEX makes use of an automatically managed memory hierarchy. It is this aspect of the project that will be of relevance in our present discussion.

As a simplistic illustration, we show in Figure 2 three levels (only) of the memory hierarchy. As can be seen, this is a multicache organization. The proposed number of caches (m) is relatively large compared to present day systems (e.g., m = 32). For the INFOPLEX



SLC : Storage Level Controller
MRP : Memory Request Processor

Figure (2)

objectives such a large number of caches is essential. It will, for example, help attain the high performance improvements sought (up to a 1000 fold increase in throughput over conventional architectures). In addition, it allows for the implementation of such features as dynamic reconfiguration and automatic recovery which are aimed at improving the reliability of the system.

Two important "boxes" in the design of Figure 2 are the Storage Level Controller (SLC) and the Memory Request Processor (MRP). The function of the SLC is to couple the local bus of a storage level to the global bus that connects all storage levels. In essence, the SLC serves as a gateway between levels. For example, the SLC of level (1) accepts requests to the lower storage levels from the caches and forwards them to the SLC of level (2). When the responses to these requests are ready, the level (1) SLC accepts them and sends them back to the appropriate caches.

The Memory Request Processor (MRP) performs such functions as: implementing the storage management algorithms (e.g., directing the transfer of information across a storage level); handling all the communication protocols that are peculiar to the particular storage modules (devices) at a storage level; and mapping virtual addresses into their real equivalents. (Note that an MRP is not needed at the cache level.)

The INFOPLEX organization described (briefly) above will constitute the context within which we shall study the different approaches for handling the multicache-consistency problem. For

further information on INFOPLEX the interested reader can consult the following references: (Madnick, 1975; Madnick, 1979; Lam, 1979; Lam and Madnick, 1979; and Hsu, 1980).

III. THREE COMMON APPROACHES FOR SOLVING THE CACHE-CONSISTENCY PROBLEM

In Part (I) we explained how a consistency problem could arise in multicache memory systems. We saw, for example, that in the two-cache organization of Figure 1(b) a word (A) that existed in both cache1 and cache2 could be modified to (V.) in cache1 and in main memory but not in cache2, and thus giving rise to an inconsistent state. This example, although rather simple, is quite adequate to demonstrate the motivations behind the basic strategies that have been used to handle the consistency problem in multicache systems. There are two such strategies. First, we could restrict the "encacheability" of data items, such that only those data items that cannot cause inconsistencies are allowed to move into the cache level. For example, words that can only be READ would be encacheable. On the other hand, all data items that could potentially cause inconsistencies are prohibited from moving into the cache level, and thus all accesses to them are done through main memory. Word (A) of the above example would, therefore, fall in this category, and so it (under this

strategy) would have been prohibited from moving into either cache1 or cache2. Thus accesses to word (A) by both CPU1 and CPU2 would have been made to its <u>single</u> copy in main memory, and no inconsistency would have resulted. The price paid, however, is that accesses to word (A) are now done at main memory speed and not at the faster cache speed.

The second basic strategy that has been employed doesn.t put any such restrictions on moving data items into the cache level. The idea here is to "invalidate" a cache line when there is a risk that its contents have been modified elsewhere in the system. When a cache line is invalidated (by setting, for example, a flag in the cache directory) it is considered not in the cache. Referring again to the above example, when the value of word (A) is modified in cache1 (and in main memory) to (V.), word (A) in cache2 is invalidated. Thus if CPU2 happens to request word (A) at a later time, it will have to access it from main memory since the invalidated version (V) in its cache is considered not to exist. CPU2 will, therefore, access the valid value (V.).

In the remainder of this section we will present more specific approaches to handle the consistency problem in cache-based systems. In particular, three approaches that are proposed in the literature will be discussed, namely, the "Broadcasting," the "Store-Controller," and the "Multics" approaches. The "Broadcasting" and "Store-Controller" approaches are based on the second strategy discussed above. They, though, implement it differently. The "Multics" approach, on the other hand, is based on the first strategy.

III.1. The "Broadcasting" Approach

The idea here (as mentioned in the second strategy above) is to invalidate a cache line when its contents is modified in another cache in the system. When a cache line is modified its address is broadcasted throughout the system so that other caches sharing the line would invalidate their now outdated version of it.

Every cache is connected to an auxiliary data path over which all other caches send the addresses of lines to be modified. Each cache constantly monitors this path and executes a searching algorithm on all addresses thus received. In case of a "hit," the affected line is invalidated.

When a CPU needs to read (or write) a word that doesn.t exist in its own cache, the word will be seized from main memory. To ensure consistency main memory must always be kept "up-to-date." This (in general) can be guaranteed only if a store-through algorithm (in which the cache and main memory are updated simultaneously) is used. As was argued before, such a restriction is not without its cost: A decrease in processing speed as store operations become limited by the speed of main memory.

Another major drawback of this approach is that the invalidation data path must accomodate a very high traffic. The mean write rate for most processor architectures lies in the range between 10 and 30 per cent (Censier and Feautrier, 1978), and thus if the number of processors is higher than two, the productive traffic between a cache

and its associated processor may be lower than the parasitic traffic between the cache and all other caches. This explains why this approach has been confined to systems with at most two caches (Censier and Feautrier, 1978).

III.2. The "Store-Controller" Approach:

The basic idea here is the same as in the "Broadcasting" approach i.e. to invalidate a cache line when its contents have been modified elsewhere in the system. It is in the implementation that the two approaches differ. Here, a "Store-Controller" SC (see Figure 3(a)) is used at the cache level to keep track of every line in every cache. The store-controller "knows," not only which lines are in which cache, but also which caches share any single line. When, therefore, a line that is shared by two or more caches is updated (i.e., modified) in one of them we do not now need to broadcast invalidation requests to all caches. We, instead, use the information in the store-controller to send invalidation requests to only those caches that are sharing the updated line, if any. In other words, the motivation behind using the store-controller is to filter out all unnecessary invalidation requests.

We will present, in a flowcharted form, an example implementation of this approach which is largely based on Tang.s proposals (Tang, 1976). In this implementation, if a processor wants to write and the line is not found in its cache, then the line is always brought to the

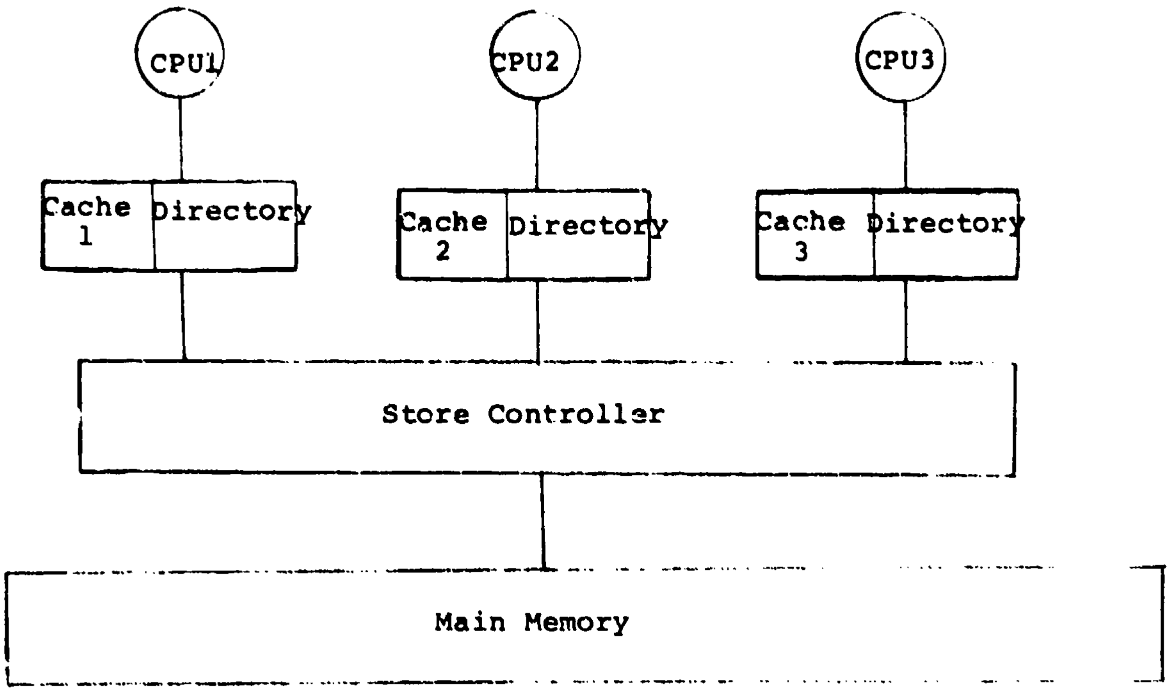
cache so that the processor can always write to cache. The store algorithm used is the "store-replacement" algorithm. This means that when a line is modified in a cache, main memory is <u>not</u> concurrently updated. It is updated later when the line has to be replaced in the cache or when other caches need to have the line.

Figure 3(a) shows how the store-controller fits conceptually into the cache organization. Figures 3(b) and 3(c) show possible layouts for the directories of both the cache and the store-controller. The "status" column of the cache directory shown in Figure 3(b) needs some explanation. The status of a line can be one of three things:

1. <u>Private</u>: For a line which has been modified (with respect to main memory) or is going to be modified. A private line exists in only one cache.
2. <u>Non-Private</u>: For a line that exists in one or more caches and which has not been modified with respect to main memory.
3. <u>Invalid</u>: For a line that has been modified elsewhere and thus becomes outdated. An invalid line is considered "not in the cache."

In Figures 4 and 5 the READ and WRITE operations are illustrated respectively.

The two major drawbacks of implementing this approach in a highly parallel multi-processor computer system such as INFOPLEX where the number of caches is relatively large are:



(a)

Main Memory Address	Cache Line Location	Status

(b) Cache Directory

Main Mem. Addr.	Cache1	Cache2	. . . .	Modified Flag
	One bit per cache is set as follows: 1 if line in cache 0 if not			Set if line is a private line in a cache

(c) Central Directory

<u>Figure (3)</u>

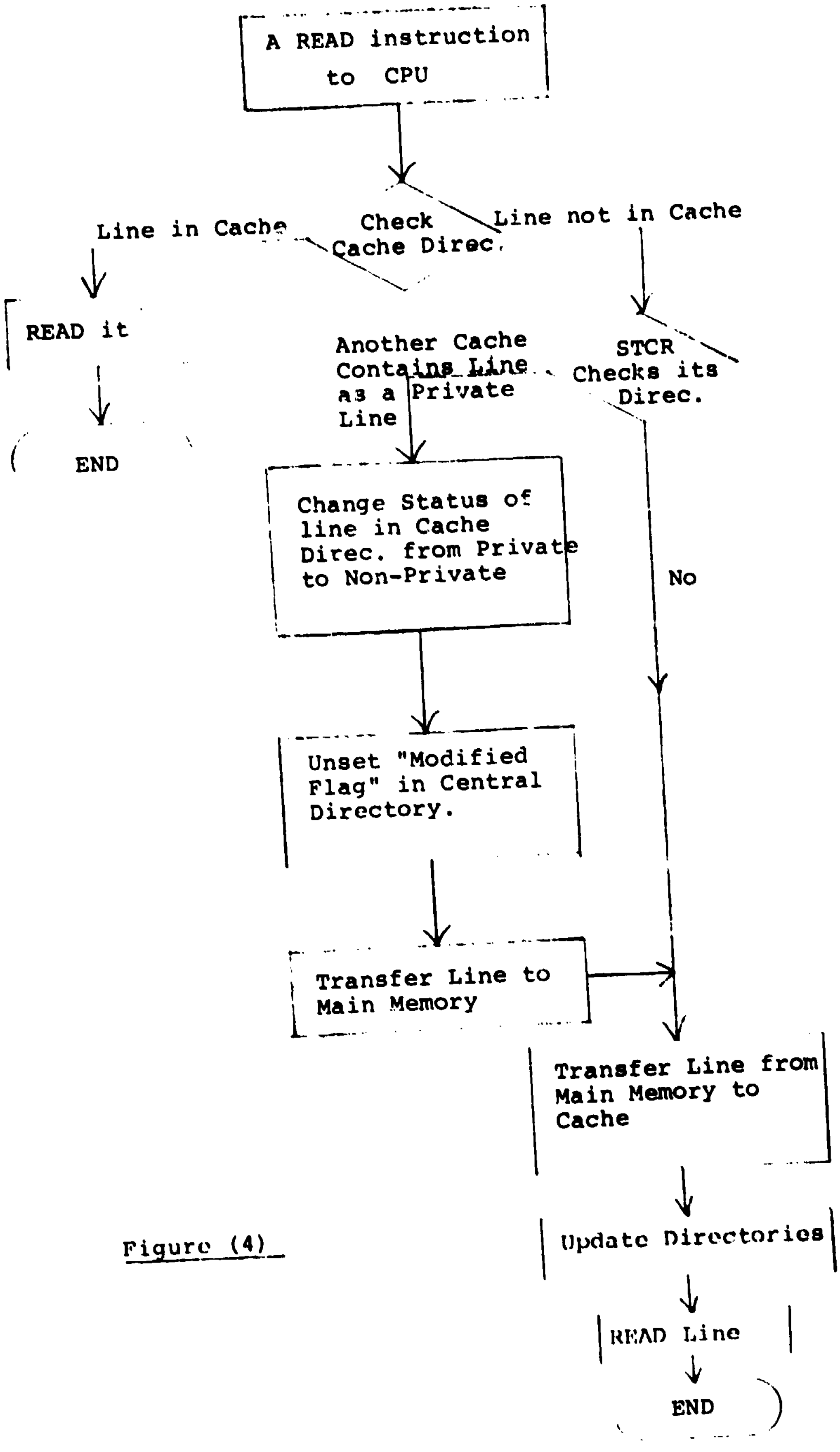

A READ instruction to CPU
Check Cache Direc.
Line in Cache
Line not in Cache
READ it
END
Another Cache Contains Line as a Private Line
STCR Checks its Direc.
Change Status of line in Cache Direc. from Private to Non-Private
No
Unset "Modified Flag" in Central Directory.
Transfer Line to Main Memory
Transfer Line from Main Memory to Cache
Update Directories
READ Line
END

Figure (4)

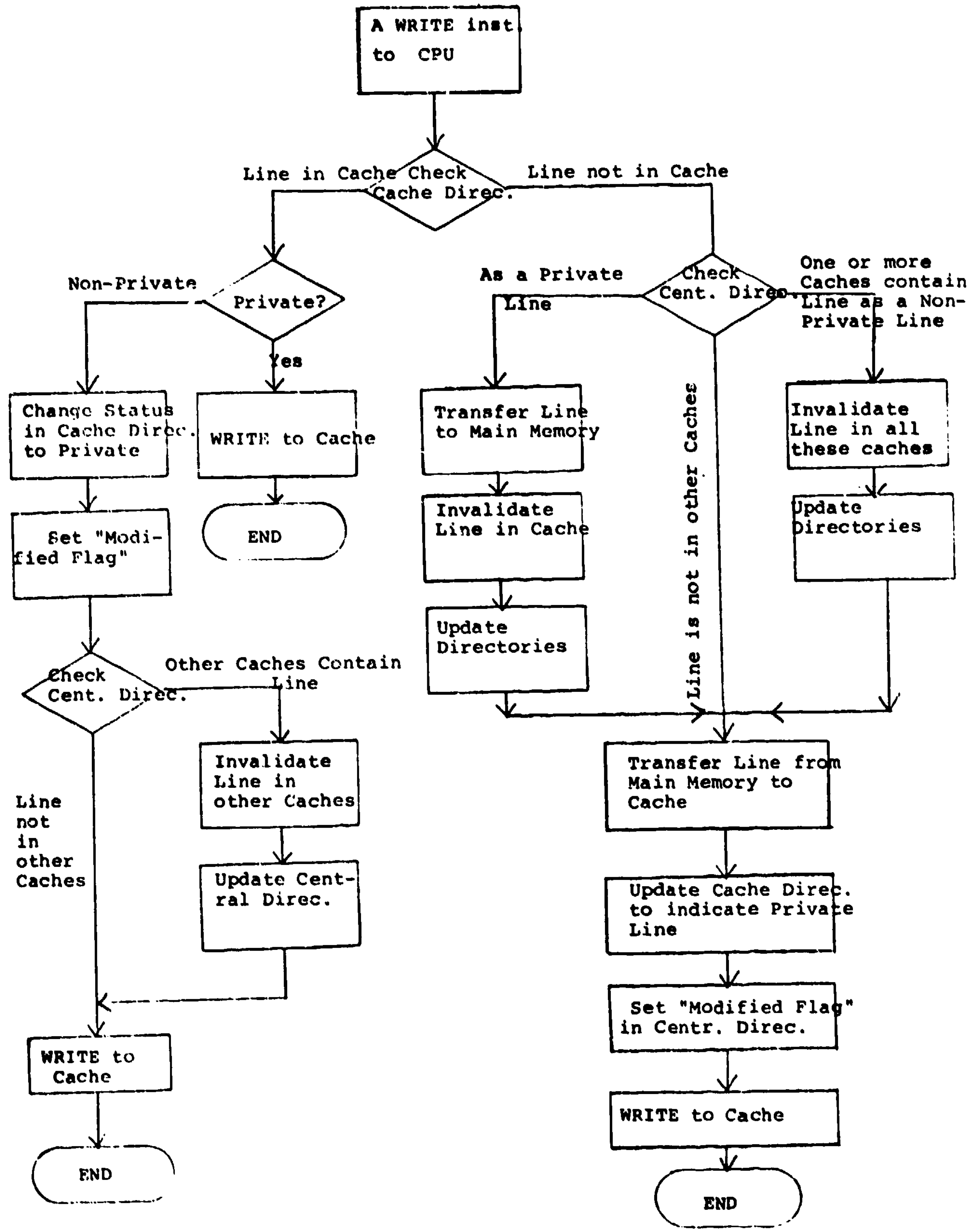

A WRITE inst. to CPU
Check Cache Direc.
Line in Cache
Line not in Cache
Private?
Non-Private
Yes
Change Status in Cache Direc. to Private
WRITE to Cache
END
Set "Modified Flag"
Check Cent. Direc.
Other Caches Contain Line
Invalidate Line in other Caches
Update Central Direc.
Line not in other Caches
WRITE to Cache
END
Check Cent. Direc.
As a Private Line
One or more Caches contain Line as a Non-Private Line
Line is not in other Caches
Transfer Line to Main Memory
Invalidate Line in Cache
Update Directories
Invalidate Line in all these caches
Update Directories
Transfer Line from Main Memory to Cache
Update Cache Direc. to indicate Private Line
Set "Modified Flag" in Centr. Direc.
WRITE to Cache
END

Figure 5

1. The size of the central directory becomes too large, which means increased time in processing it and increased costs in building it.

2. The store-controller could become a bottleneck in the system as the traffic between itself and the caches becomes very large.

III.3. The "MULTICS" Approach

This approach, which is used in Honeywell.s Multics computer system (Greenberg, 1978), has two important features. Firstly, the Multics cache is a "store through" cache. This means that the cache and main memory are updated simultaneously. The second feature is that the system address space is divided into segments, each of which has associated with it names and per-user access rights which govern the ability of each potential user of the segment to read and/or write its contents (i.e., words).

Every segment is known by the system to be either "writable" or "non-writable." The non-writable class of segments, that is those to which no users have write access, is an important and statistically significant one in MULTICS. All procedures, including all parts of the operating system, utilities, libraries, translators, and so forth, fall into this category. These segments are "encacheable" by all processors. This means that their contents (or words) are allowed to "migrate" to any or all caches. Notice that when such words find their way into a cache (or more than one cache) there is no possibility for a

consistency problem to arise (for such words) since no processor can write or modify them.

For the other class of segments, those which are writable, the situation is slightly more complicated. For a segment falling in this category, there are three possible states:

1. One or more processes (users) are accessing the segment but none of them has a write access to it. The segment is encacheable by all processors but no consistency problem will arise.
2. One or more processes are accessing the segment, with at least one of them having write access. The segment becomes non-encacheable i.e., its words cannot migrate to any of the caches. The consistency problem will not arise here also since there will only be one copy (i.e., the one in main memory).
3. Only one process that has write access is accessing the segment. The segment is encacheable only to that process. And it is only in this third state that the consistency problem could possibly arise. Consider the following scenario:

> Assume a certain segment S1 is addressable by only one process PROC1 which is currently running for the first time on processor CPU1. Assume also that PROC1 has write access to S1. Thus S1 is encacheable by CPU1. As long as CPU1 runs PROC1, words of S1 may be drawn into CPU1.s cache and be modified by CPU1 with no problem. During this period, no other processor can address S1, for by assumption it is addressable only by PROC1, which is uniquely associated with

CPU1 during the interval in question. Thus, it is impossible for other processors to draw words of S1 into their caches as long as PROC1 is associated with CPU1. Until CPU1 leaves PROC1, there is thus no danger of words of S1 in CPU1.s cache becoming outdated, as no other processor can address S1. Similarly, there cannot be words of S1 in any other processor.s cache, for by assumption, CPU1 was the first and only processor to run PROC1. Thus, there is no danger that modifications to words of S1 made by CPU1 can invalidate copies in other processors. caches, since such copies cannot exist. Potential difficulty arises when CPU1 has left PROC1, and some other processor attempts to run PROC1. The first time this happens, there is no problem. Since all words in main memory are accurate, by virtue of the store-through cache, another processor, say CPU2, cannot have inaccurate data, or main memory is accurate, and we have just shown how CPU2.s cache may not contain inaccurate data. However, while PROC1 runs on CPU2, CPU2 may modify words of S1 in its own cache and in main memory. Still there is no problem. Main memory is accurate, as is CPU2.s cache. This can go on like this as long as processors which have never ran PROC1 (since PROC1 started running) run it. However, the first time some processor which has already ran PROC1 since then attempts to run it again, the scheme appears to break down. Assume CPU1 attempts to run PROC1 for the second time. There may be words of S1 in CPU1.s cache from the previous time CPU1 ran PROC1. Some of these words may have been modified by PROC1 while it ran on CPU2. Thus, these words are accurate in main

memory and in CPU2.s cache, but are inaccurate in CPU1.s cache, for CPU2 had no way of knowing or acting upon the fact that they were in CPU1.s cache.

The MULTICS solution to "its" consistency problem is simple: clear the cache of a processor upon entering a process if it was not the last processor to run that process. This is performed by the MULTICS process dispatcher, with a special processor instruction that accomplishes this task. This ensures that no words of any per-process writable segment will be found in a processor.s cache if there was any possibility that any of those segments may have been modified by other processors. The operating system maintains in the control block describing each process the identity of the last processor to have run this process thus this check is easy to make when a processor is dispatched into a process.

From an INFOPLEX-type-system view point, there are three major drawbacks to the MULTICS approach, all of which are performance related:

1. The class of segments that are non-encacheable can be of significant size, and thus dampening the performance gains sought by the cache organization. Note that in MULTICS there is the significant class of segments which we termed "non-writable" and which are encacheable. In data base computers, like INFOPLEX, this category which contains things like utilities, libraries, and translators will probably be much smaller, and thus decreasing the portion of encacheable segments. (There could, however, be

special applications where this doesn.t apply e.g., READ-only data base applications.) In addition, the MULTICS approach doesn.t discriminate between two processes who although both have write access to a segment, one actually exercises the "right" and modifies the segment, while the other doesn.t. In both cases the segment will become non-encacheable if there are other processes that are also reading it. This means that in both cases the system.s speed will be slowed down to the speed of main memory.

2. Using a store-through algorithm has its own performance disadvantages. As was stated earlier, it limits the speed of store operations to that of main memory, and thus defeating the very purpose of using a cache.

3. The procedure of clearing up the cache is also a wasteful one. Note that a cache is always cleared if its processor wasn.t the last one to run the process. All access requests to the cleared cache contents that would have otherwise been serviced by the cache, must now wait for transfers from main meemory. This will, obviously, slow down the system.

III.4. Conclusion

We have analyzed the three common approaches that have been proposed in the literature to handle the consistency problem in multicache systems. We have considered each approach in the context of the INFOPLEX framework presented earlier. Within this context we were able to identify some major drawbacks in each of the approaches.

In the next section we propose a new approach to handle the cache-consistency problem in multi-processor architectures. In Part (V) we will evaluate the performance of this proposed approach.

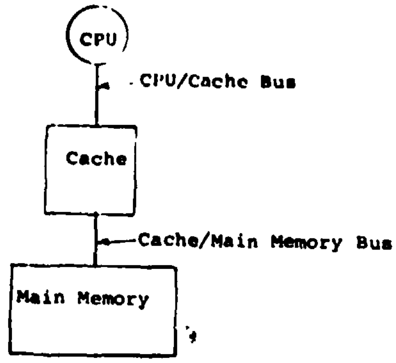
IV. THE "COMMON-CACHE / PENDED TRANSACTION BUS" APPROACH

IV.1 Introduction

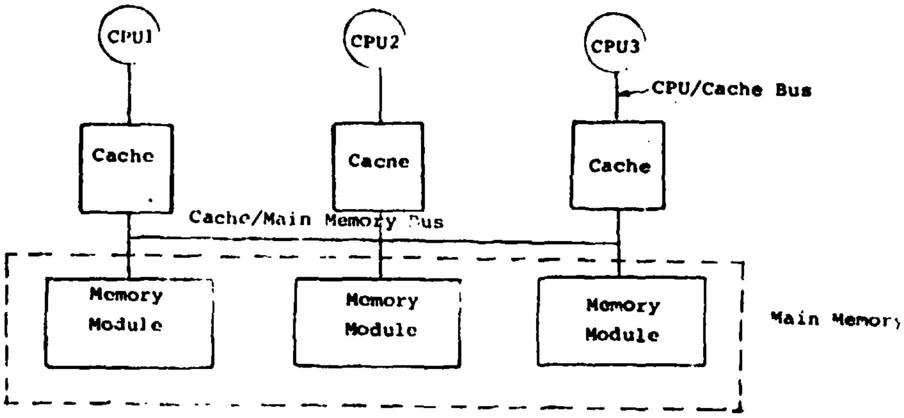
We argued in the beginning of Part (I) that in a single-cache memory system (Figure 6(a)), where there is only one access path between each level, no cache-consistency problem will arise. Once two or more caches are used, however, the potential for the problem develops.

The "traditional" approach in employing caches in multi-processor systems has been to basically replicate the structure of Figure 6(a) for each of the processors, as shown in Figure 6(b), and then solve any problems that arise. A problem that arises, of course, is the cache-consistency problem, and the three basic approaches that have been developed to handle it are those of Part (III).

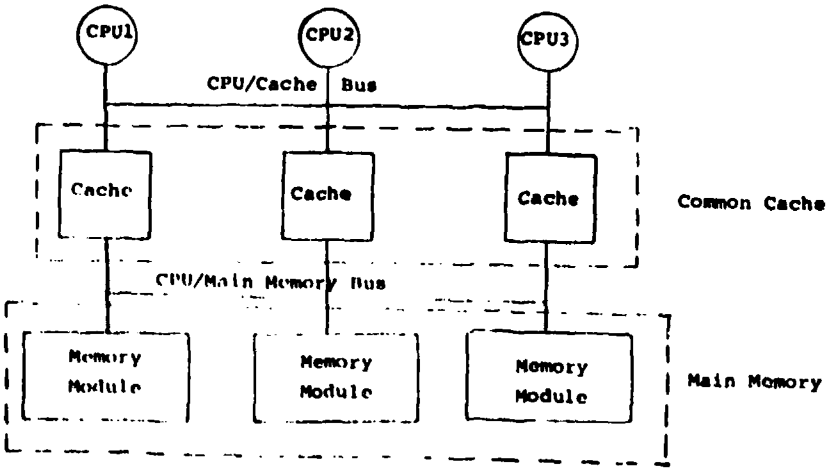
Implementing any of the three approaches will obviously constitute some processing overhead. As an example, consider the



(a)



(b) Private Caches (PC)



(c) Common Caches (CC)

Figure (6)

"Store-Controller" approach and refer in particular to the flow-chart of Figure 5. When a CPU needs to modify a line that exists in its cache, and the line happens to be a "non-private" line, then the status of the line is first changed to "private" and the "modified flag" is set. Next, the Store-Controller.s directory is checked, and if the line is found not to be shared by other caches it is modified. If, however, it happens to be shared, then messages are sent to the appropriate caches to invalidate the line, the central directory is updated, and finally the line is modified. How much overhead did we incur to maintain consistency? Well, compare the above steps with those needed for a uniprocessor system where the cache-consistency problem does not arise. In such a system, when the CPU needs to modify a line that exists in its cache, it simply proceeds and modifies it. None of the above checks, updates and messages are needed.

The concern over the processing overhead needed to maintain consistency is a legitimate one. Such overhead does undoubtedly dampen the performance gains (e.g., system throughput) which are sought by introducing caches in the first place. Attempting to minimize this overhead, therefore, seems an attractive direction for research work.

In this research endeavor, we are basically proposing an architecture that eliminates the processing overhead associated with handling the multicache-consistency problem. The architecture is depicted in Figure 6(c). The important distinction between this organization and that of Figure 6(b) is that, here, each of the cache modules is accessed by, and thus services, all of the processors. Thus, just as main memory is common to and shared by all processors,

the cache level in our proposed architecture is also common to and shared by all processors. In such a scheme there is no need to store more than a single copy of any data item at the cache level. This eliminates redundancy. And with no redundancy at the cache level, no inconsistencies can obviously arise, which in turn eliminates the need for mechanisms to maintain cache-consistency and the overhead associated with such mechanisms.

It is important to note that this architecture preserves the basic intent behind cache-based systems, namely, using a high speed memory level between the CPU and main memory in order to bridge the speed gap between the two. It, however, introduces the concern as to whether the CPU/cache bus (see Figure 6(c)) can handle the needed high volume of traffic. Comparing the Private Cache (PC) architecture of Figure 6(b) against the Common Cache (CC) architecture of 6(c), we note that the CPU/cache bus in PC handles the transaction load generated by a single processor where as in CC it handles the load generated by all the CPUs. We can, therefore, expect that the load on the CPU/cache bus in our proposed (CC) architecture to be close to N-times that of the PC architecture, where N is the number of processors. Of course, the load will be somewhat less than exactly N-times because in PC some overhead traffic will be generated by the mechanism used to handle the cache-consistency problem, and which will not be needed in (CC).

Thus, to re-state, the Common Cache approach eliminates the cache-consistency problem (i.e., by eliminating private caches) but there is a fundamental concern that the CPU/cache bus will develop into a bottleneck that degrades the system.s performance. In the next

section we introduce the Pended Transaction Bus Protocol, which we believe provides the basis for a viable solution to the problem.

IV.2 The Pended Transaction Bus (PTB)

The degree of bus utilization of a processor is a function of the physical characteristics of the bus, and the protocol used on it. The physical characteristics of the bus which include the length, voltage levels, impedence, termination, capacitance, noise immunity, and overall reflection characteristics, affect its operating speed. Ultimately, any bus is limited by the speed of electricity along a conductor (0.6 to 0.9 nanoseconds per foot typically, depending on wire characteristics). Careful electrical analysis and physical layout can optimize these parameters to achieve reasonable electrical speed.

Given an electrical bandwidth of the bus, as formed by the bus wires and the driving logic, the actual data bandwidth becomes a function of the protocol used on it. The traditionally high bus utilizations of multi-microprocessor architectures that employ the single bus as their interconnection scheme is primarily a result of the bus protocol used, and which is called the "master-slave" protocol. In such a protocol, the CPU (master) asserts a request on the bus, and the memory (slave) that receives it does the appropriate action (e.g., a READ), and then responds (with the requested data). The bus is viewed as "busy" during this entire time. A large portion of this time is actually spent waiting for the slave to complete the requested action.

The electrical and logical time to transmit the actual request and return the reply are a relatively small portion of the total bus usage time. The period of time between the request and the acknowledge is a wait interval, and no useful work is done with the bus during this time. Bus utilization could be significantly reduced if we released the bus during the wait interval. To do this we split the transaction into two parts, a request part and a reply part. The master requests the bus, and upon being granted it, sends the request to the slave at maximum speed. The slave acknowledgles reception of the request, and starts to work on it. The processor then releases the bus and waits for the results. When the slave completes its task, it asks for the bus, and upon receiving it sends the reply back to the originating master at maximum speed. The time between the request and reply can be used by other masters and slaves to transfer other messages over the bus. Because the slave stores the incompleted request from its master, it is called a pended transaction, and the bus protocol, developed at M.I.T., is called a Pended Transaction Bus (PTB) protocol (Toong, et al, 1980).

In the above discussions, it was presumed that the slaves were always able to accept the master requests when presented. This would imply that each master was using different slaves to guarantee such separation. Given the shared nature of the cache data, it is likely that two (or more) masters may make requests to the same cache-memory slave. The simplest scheme to resolve this contention problem is for a busy slave to refuse a new request and make the requesting master retry at a later time when the slave becomes free. This, however, is wasteful of bus bandwidth, since the time spent to send a request out

the first time, only to be refused, is not useful. Furthermore, the slave may still be busy when the master tries again later.

Goodrich (Goodrich II, 1980) has proposed putting queues on the inputs of the slaves to buffer requests. That is, any requests that come while the slave is busy would be placed in a first-in-first-out queue, and these requests would be serviced in order when the slave is able to handle them. Such a scheme would reduce the bus load to what is actually needed for transmission, without any extra cycles. In addition, it would also improve the slave response time as seen by the master over the simple scheme described before, since the slave would have the request in its queue, and would service it as fast as it could, not just when the master is finally successful in transmitting it. Note that a queue overflow need not be fatal in this system. It can be treated like a refusal in the previous scheme, and would simply require the master to retransmit the request later. If the queue size is sufficiently large, such refusals would be rare. Finally, for best slave throughput, there should also be a queue on the output of each slave.

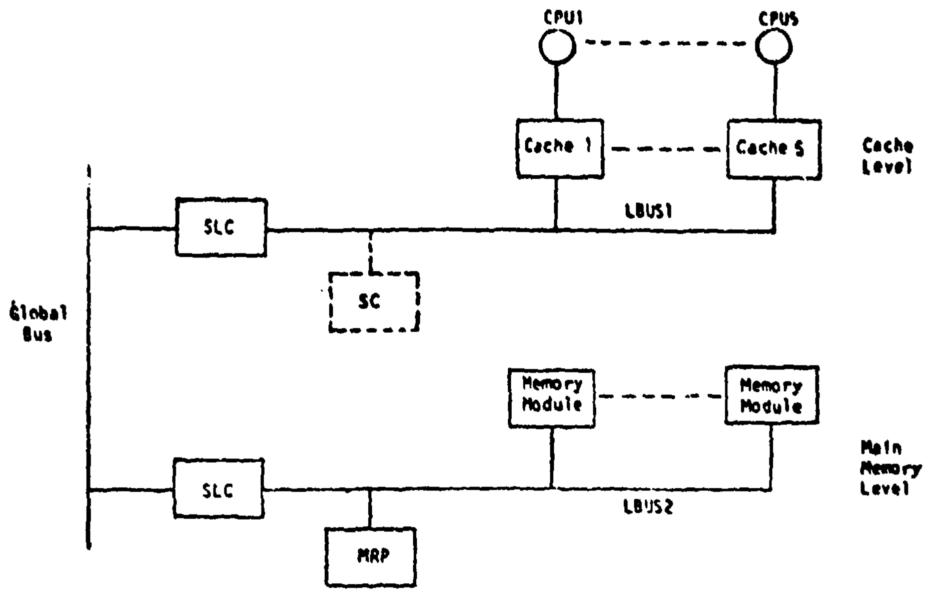
IV.3 An Application: The INFOPLEX Storage Hierarchy

In the above sections we have introduced an approach to the cache-consistency problem in highly parallel multi-processor computer systems, which we will call the "Common-Cache / Pended Transaction Bus" approach (CC/PTB). Its two distinctive features are: Firstly, the

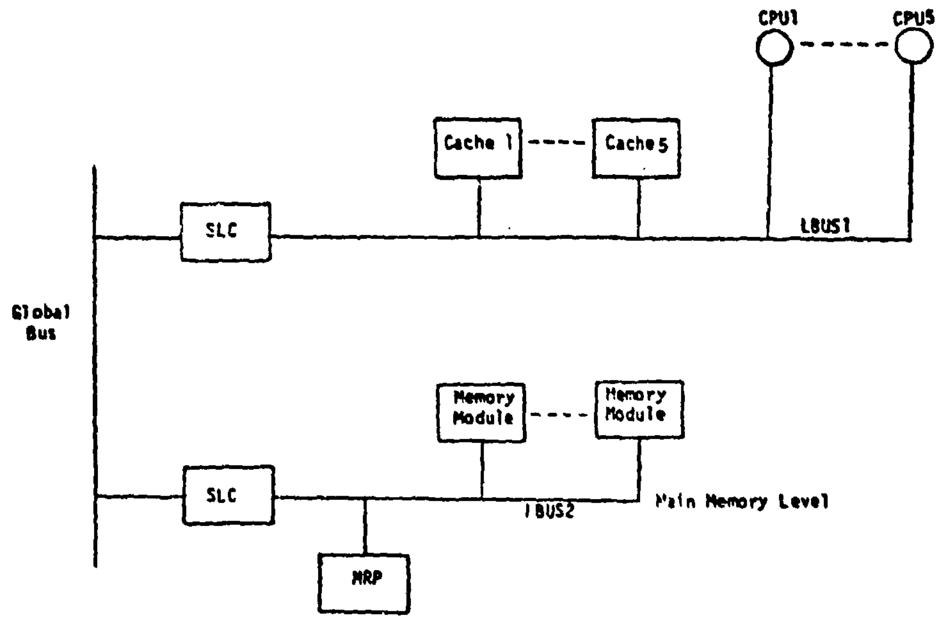
conventional one to one relationship between processors and caches i.e., where each CPU has its own private cache, is replaced by an N-to-M relationship, where a pool of cache-modules is commonly shared by all processors. Secondly, we propose the use of the Pended Transaction Bus (PTB) as the interconnection scheme that connects the processors and the cache-modules.

In this section we describe how the CC/PTB architecture can be incorporated into the storage hierarchy of the INFOPLEX data base computer. Schematically the proposed architecture would look like Figure 7(b). We will, henceforth, refer to this architecture as INFOPLEX/CC (for Common Cache) while referring to the original Private Cache organization (shown in Figure 7(a)) as the INFOPLEX/PC architecture.

The key INFOPLEX storage hierarchy operations are the READ and WRITE. In INFOPLEX/PC two strategies are reeded to implement these operations, a strategy for the cache level and another for all other levels. The reason for this is manifested in Figure 7(a), where it can be seen that the organization of the cache level is different from the other levels of the hierarchy. In INFOPLEX/CC, on the other hand, the cache level organization is very similar to that of the lower storage levels. As a result, the strategy used to implement the READ and WRITE operations could be the same for all levels of the hierarchy with very minor provisions to account for the few differences that do still distinguish the cache level. (For example, the absense of an MRP at the cache level.)



(a) Private Cache (PC) Architecture



(b) Common Cache (CC) Architecture

LBUS1: Local BUS at level (1) - cache level

LBUS2: Local BUS at level (2) - main memory level

SC: Storage-Controller - needed only if "Storage-Controller" approach is implemented

SLC: Storage Level Controller - it couples a storage level to the Global bus (i.e. serves as a gateway between levels)

MRP: Memory Request Processor - performs address mapping function

Figure (7)

We will attempt now to explain briefly how the basic READ and WRITE operations will be implemented in the INFOPLEX/CC architecture. To a large extent this will be based on the work of Lam (Lam, 79), where a much more detailed and complete discussion is presented.

All READ and WRITE operations are performed in the highest storage level L(1) i.e., the cache level. If a referenced data item is not in L(1), it is brought up to L(1) from a lower storage level via a READ-THROUGH operation. The effect of an update to a data item in L(1) is later propagated down to the lower storage levels via a number of STORE-BEHIND operations.

When a READ request is issued by a processor, the cache level is checked to see if the requested data is in it. If the data is found in a cache-module, it is retrieved and returned to the processor. If, however, the requested data is not found in the cache level, a READ-THROUGH request is queued to be sent to the next lower level L(2) via the Storage Level Controller (SLC). As mentioned in Part (II) the SLC serves as a gateway between the storage levels of the hierarchy.

At a storage level, a READ-THROUGH request is handled by the Memory Request Processor (MRP). An MRP performs the address mapping function. It contains a directory of all the data maintained in the storage level. Using this directory, the MRP can determine if the requested data is in one of the storage devices at that level. If the data is not in the storage level, the READ-THROUGH request is queued to be sent to the next lower storage level via the Storage Level Controller (SLC).

If the data is found in a storage level L(i), the MIP maps the main memory address of the requested data item into its real address in L(i). This real address is used by the appropriate storage device to retrieve the block containing the data and then passes it to the SLC. The SLC would then broadcast the block to all upper storage levels by dividing it into fixed size packets. Each upper storage level has a buffer to receive these packets. A storage level only collects those packets that assemble into a sub-block of an appropriate size (peculiar to the storage level) that contains the requested data. This sub-block is then stored in a storage device. At L(1), the sub-block (i.e., the cache line in this case) containing the requested data is stored, and the data is finally sent to the processor that initiated the request.

In a WRITE operation, the data item is written into a cache-module, and the processor is notified of the completion of the WRITE operation. We shall assume that the data item to be written is already in L(1). (This can be realized by reading the data item into L(1) before the WRITE operation.) A STORE-BEHIND operation is next generated by the cache-module and sent to the next lower storage level. INFOPLEX uses a two-level STORE-BEHIND strategy. This strategy ensures that in a hierarchy with N levels, an updated block will not be considered for eviction from a storage level L(i), until its "parent" blocks at levels L(i+1) and L(i+2) are updated. This scheme will ensure that at least two copies of the updated data exist in the storage hierarchy at any time. The motivation behind using such a strategy is two-fold. Firstly, the reliability of the system is enhanced because at least two copies of newly written data are always maintained until the data is "securely" copied at the lowest level of

the hierarchy. Furthermore, the STORE-BEHIND strategy allows for the updating of lower storage levels to be carried out at slack periods of system operation, thus enhancing performance.

In the above discussions we didn.t show how a specific cache-module can be correctly selected to handle a READ or a WRITE operation of a particular data item. In all but the cache level this function is performed by the Memory Request Processor (MRP), which maps main memory addresses into their physical equivalents at a particular level. What we need, therefore, is to augment the processors/common-cache interface to translate main memory addresses to physical addresses in the cache-modules.

There are two possible places to perform the translation operation: at the processor interface and at the cache interface. At the processor interface, the address gets translated before it reaches the bus. This requires that each processor be "informed" about all current lines at the cache level. This information would be used to translate all the main memory addresses of these lines to their equivalents at the cache level. An advantage of this is that the translation can be performed while the procesor is arbitrating for the bus, and if the translation operation is fast enough, it can be done without any access time penalty.

In the second possible scheme the address gets translated at the cache-module interface. What would be needed here is a mechanism incorporated in the recognition circuitry of each cache-module that would translate the main memory address put on the bus, and that would

accordingly decide IF the desired data item is present in the cache level and if so WHERE i.e. in which cache-module and in which location in it. Both these should be done as fast as possible. Tag directory schemes incorporating set associative mapping are suggested by Mattick (Mattick, 1977).

In Part (V) we will evaluate the performance of both these schemes when implemented at the cache level of the INFOPLEX storage hierarchy.

V. EVALUATING THE PERFORMANCE OF THE "COMMON-CACHE / PENDED TRANSACTION BUS" APPROACH

V.1. Introduction

To evaluate the performance of our CC/PTB approach we used the INFOPLEX multi-level storage system as our test case. Very slight modifications to the existing design were needed to incorporate "CC/PTB" into the INFOPLEX storage stucture. For example, instead of using the four level memory hierarchy studied previously by Lam (Lam, 1979a) just two levels, the cache level and main memory were sufficient for our purposes.

The "CC/PTB" approach was compared against two benchmarks. Firstly, we evaluated the performance of the "Store-Controller" approach as a representative of the traditional approaches. Selecting the "Store-Controller" approach to evaluate our scheme against cannot, however, provide an effective answer to the question of how "optimal" either approach is. What is needed is an "absolute" benchmark against

which both approaches could be judged. For this purpose we evaluated the performance of a traditional private cache architecture assuming no overhead for handling the cache-consistency problem. This, then, constitutes the "performance ceiling" that no cache-consistency handling mechanism (for INFOPLEX) could possibly exceed. It is also a valuable reference point in that it tells us how close we are to an "optimal" solution.

V.2. The Evaluation Tool

The evaluation was performed by producing a simulation model in GPSS. We developed four separate GPSS programs:

1. Program "OPT" ignores the cache-consistency problem, and thus incorporates no overhead for handling it. It provides us with a ceiling on performance.
2. Program "STCR" incorporates the "Store-Controller" approach.
3. Program "CC/PTB/C" incorporates the "CC/PTB" approach with the translation operation performed at the cache-interface.

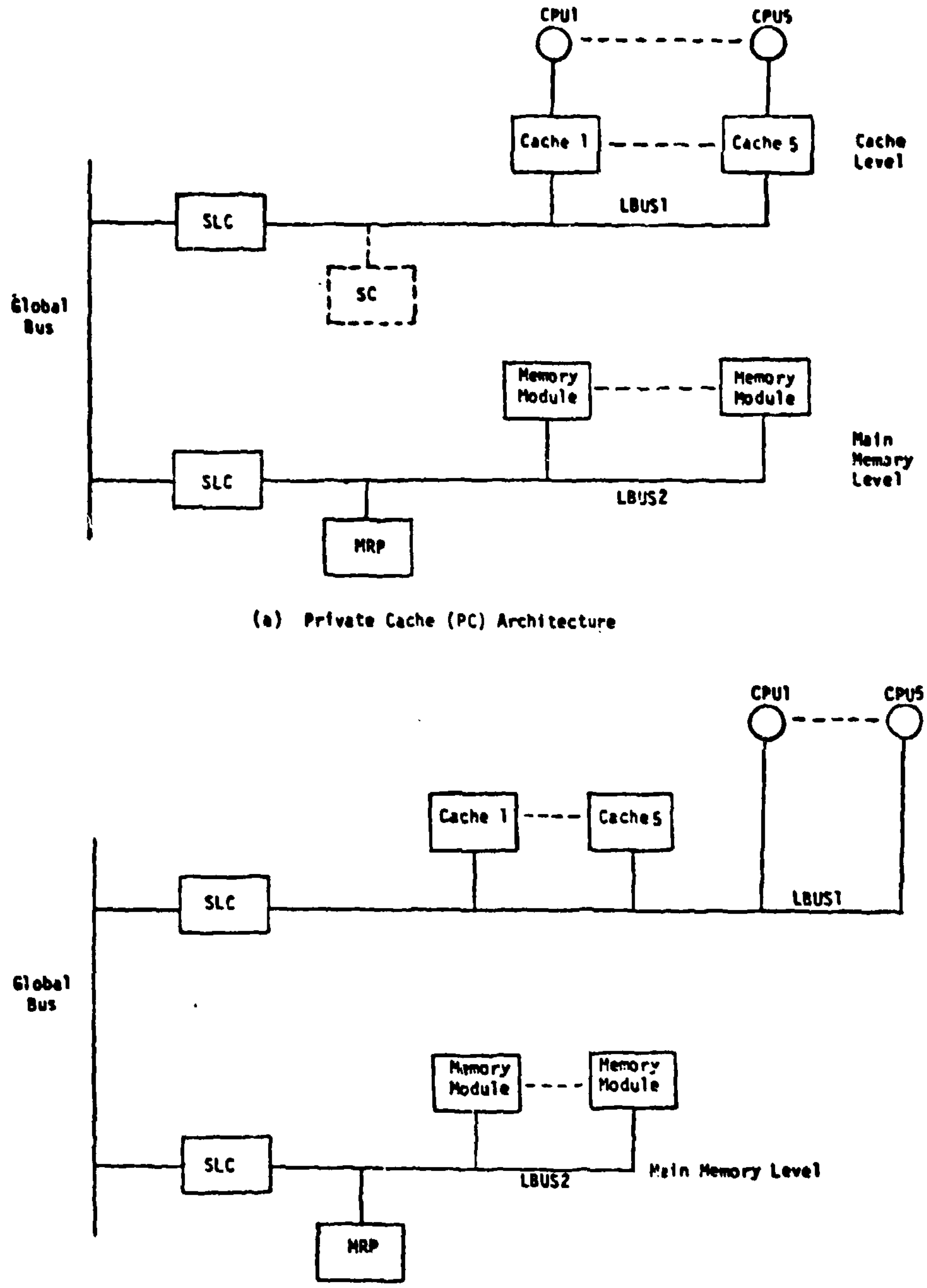
and 4. Program "CC/PTB/P" incorporates the "CC/PTB" approach with the translation operation performed at the processor interface.

The GPSS code for each of the above four programs is presented in a separate Appendix (Appendices I through IV). All four programs are highly detailed simulators. A widely used index that characterizes the degree of detail of a simulation model is its resolution in time, which

is defined as the shortest interval of simulated time between two consecutive events being considered (Ferrari, 1978). In our four simulations, the resolution in time is 10 nanoseconds. Such detailed simulators are likely to be more accurate and to have broader field of application than less detailed ones. However, they are certainly more expensive to design, implement, test, document, and use.

In addition to deciding on the degree of detail of the models, another basic decision had to be made concerning the workload that would drive the simulations. We decided to perform our measurements in an operating environment that is not at all uncommon in present-day computer systems, namely, running at capacity. Under such conditions, there will always be at least one transaction in the system.s input queue(s) waiting to be serviced. In such a case, the performance characteristics that are of interest to us, such as throughput per unit time, become insensitive to the distribution of job arrivals.

Before concluding this section we would like to emphasize some of the structural differences between the four models, as well as some of their common characteristics. The basic structural differences are highlighted in the diagrams of Figure 8. In Figure 8(a) a two-storage-level version of the INFOPLEX storage hierarchy proposed in (Lam, 1979b) is shown. To support the "Store-Controller" approach an SC "box" (dotted in Figure 8(a)) is added to the architecture. In Figure 8(b) the architecture we proposed in Part IV to support the two versions of the "CC/PTB" approach is incorporated into the INFOPLEX storage system. Notice that in the CC/PTB architecture we deliberately maintained the same number of caches (5) as in Figure 8(a). It is



(a) Private Cache (PC) Architecture

(b) Common Cache (CC) Architecture

LBUS1: Local BUS at level (1) - cache level

LBUS2: Local BUS at level (2) - main memory level

SC: Storage-Controller - needed only if "Storage-Controller" approach is implemented

SLC: Storage Level Controller - it couples a storage level to the Global bus (i.e. serves as a gateway between levels)

MRP: Memory Request Processor - performs address mapping function

Figure (8)

important to realize that while a 1:1 relationship between the CPUs and the caches is inherent in the Private Cache (PC) architectures, such is not the case for the Common Cache (CC) architectures. We, however, chose to maintain the same number of caches in both architectures (and four simulation models) to neutralize it as a factor that might affect performance.

Finally, there is a set of common characteristics that are shared by all four models, these are:

Degree of Multiprogramming of a CPU	= 10
Bus Width	= 8 bytes
Size of Transaction without data	= 8 bytes
Size of Transaction with data:	
at Level 1	= 8 bytes
at Level 2	= 64 bytes
Size of Data Buffers	= 10 64-byte transactions

Finally, the Pended Transaction Bus (PTB) protocol will be used in all three architectures (and four models). That is, we are committed to the PTB protocol in INFOPLEX as a result of our experimental work (at M.I.T.), which demostrated its performance advantage.

V.3. The Simulation Experiment:

Our criterion for measuring performance in this experiment, and which will also serve as our dependent variable, will be the total system throughput as measured by the total number of transactions processed per unit of time. As for the independent variable, there

were two candidates: (1) the hit ratio, which is the percentage of time that a referenced data item is found in the cache level; and (2) the transaction mix i.e., the percentage of READ requests versus WRITE requests.

The latter was chosen because we felt it is in a sense, a more independent variable. What we mean is that the transaction mix is largely a function of the use of the system and as such we have very little control over it. The hit ratio, on the other hand, is system-dependent. It can be affected by manipulating such system parameters as the total size of the cache level and the size of the individual cache blocks.

Thus, the hit ratio will be held constant, throughout the experiment, at a value of 0.90 for READ requests and 1.00 for WRITE requests. (That is, we are assuming that a WRITE of a data item is always preceded by a READ to it.) The transaction mix i.e., the percentage of READs, will be allowed to vary in the range from 70 % to 90 %. This is the range in which the mean READ rate lies for most processor architectures (Censier and Feautrier, 1978).

Both the hit ratio and the transaction mix are variables that affect the performance of the system but which are independent of the mechanisms used to handle the cache-consistency problem. There are, however, variables that are peculiar to the particular mechanism used, and which influence the system.s performance significantly.

In the "Store-Controller" approach the degree of sharing between

the caches is by far the most important such variable. We evaluated the "Store-Controller" approach under two operational modes, a "pessimistic" mode and an "optimistic" mode. Under the optimistic mode no sharing between caches takes place, yielding an upper bound on the performance of this approach. Under the pessimistic mode a high degree of sharing will be introduced (50 % of the cache blocks will be shared by more than one cache-module). This will then provide us with a conservative lower bound on the performance of the "Store-Controller" approach.

With respect to the "CC/PTB" approach we will also follow the above strategy, and evaluate it under both "pessimistic" and "optimistic" conditions. Under the optimistic condition we will assume the load on the cache-modules to be uniformly distributed (i.e., each of the 5 cache-modules carries 20 % of the load). And for the pessimistic case we will assume the load to be linearly distributed between 10 % at the least loaded cache-module and 30 % at the most loaded.

V.4. The Hardware Parameters

It is necessary to determine the speeds of the different hardware components in the INFOPLEX storage hierarchy. In particular, what we sought were the best possible 1985 projections for these speeds, since the first hardware prototype of the INFOPLEX data base computer was not expected before then.

The forecasts shown below constitute a realistic, but ambitious, scenario for 1985. In other words, they incorporate the fastest possible components that we envision as being available (and appropriate) for building an INFOPLEX in 1985. (Only a subset of the parameters are shown.) The bus speed (b) is 10 nanoseconds, the cache READ/WRITE speed (c) is 20 nanoseconds, and the remaining parameters are multiples of the latter, as shown. In other words, we used the cache speed as a logical building block to "build" the forecasts of the other hardware components (other than the bus). For example, if the cache READ/WRITE speed is 20 nanoseconds the READ/WRITE speed of main memory would be 10 x c = 200 nanoseconds. The parameter values are:

Bus speed (b)	= 10	nanoseconds
Cache READ/WRITE speed (c)	= 20	"
Directory Lookup (2c)	= 40	"
Directory Update (4c)	= 80	"
Storage Level Controller (SLC) speed (2c)	= 40	"
Main Memory READ/WRITE speed (10c)	= 200	"

Three other scenarios will be tested. The bus speed, in all three, will remain at 10 nanoseconds. The cache READ/WRITE speed, however, will take the increasing values of 40, 60, and 80 nanoseconds. And finally, the remaining parameters will maintain their relative values in terms of n, the cache READ/WRITE speed. For example, when the cache READ/WRITE speed (c) becomes 40 nanoseconds the READ/WRITE speed of main memory will be 10 x c = 400 nanoseconds.

Selecting several "good" 1985 scenarios reflects the fact that different options, in building an INFOPLEX, will be available to accomodate the cost/performance tradeoffs. And because the bulk of the

system.s cost lies largely in the storage components, the variations in the forecasts between the different scenarios involved mainly those components.

V.5 Analysis of the Simulation Results:

As mentioned previously, our dependent variable in this experiment is system throughput. It is also the criterion we use to evaluate the performance of the cache-consistency handling mechanisms.

The throughput of any computer system is bounded by one of two factors, namely, bottlenecks in the system or the transaction load on it. In section V.2 we mentioned that our models will operate in a maximum load closed-loop environment, where new transactions are continuously generated to replace serviced ones. In such an environment, bottlenecks will definitely arise, limiting the throughput of the system.

Thus, in the process of interpreting the simulation results, we wish to identify and analyze system bottlenecks. In such an analysis, one needs to consider four major factors that directly influence the evolution of a particular system component into becoming the system.s bottleneck. The four factors are:

1. Architecture: Consider for example the (a) PC and (b) CC architectures of Figure 8. In PC the local bus of level 1 (the

cache level) handles only the communications between the five caches and main memory. In the "CC/PTB" architecture the cache level bus must handle, in addition, the communications between all the processors and the cache-modules. The chances for the local bus of level 1 (LBUS1) to become the system bottleneck are, therefore, much higher in the CC architecture than they are in PC.

2. Algorithms and Protocols: The set of algorithms supported by an architecture and the protocols and mechanisms used to implement them undoubtedly influence the utilization patterns of the different architectural components. Consider for example the central role of the Store-Controller "box" in implementing the algorithms that support the READ and WRITE operations in the "Store-Controller" approach. Such a role will inevitably lead to high levels of utilization of the Store-Controller.

3. Workload: We mentioned in section V.3 that we intend to evaluate the "CC/PTB" approach using two different load distributions on the cache-modules, a uniform distribution and a linear one. In the latter case, the cache-module carrying the highest load (i.e., 30 % of total load) could clearly develop into a system bottleneck.

4. Hardware Components. Characteristics: The characteristic of significance here is speed. For an example we refer again to Figure 8. All communications between level 1 and level 2 in the INFOPLEX storage hierarchy go through the Storage Level Controllers (SLCs) of both levels as well as through the Global Bus. The time needed by the Global Bus to process any of the communication messages (i.e., the time it takes to transmit the message) will usually be less than that needed by the SLC to

process the same message. This means that the SIC will always saturate before the Global Bus can develop into a bottleneck.

As part of the standard GPSS output, the utilizations of all hardware components (that are modelled as GPSS "facilities") are printed. This allows us to precisely identify the system bottleneck(s). An example is shown in Figure 9. In this case the bottleneck is LBUS1 (the local bus at level (1)) with a utilization of approximately 100 %.

In Figure 10 our simulation results for the four scenarios are presented. In the discussion that follows, we will identify the different scenarios by their cache speed/bus speed ratios (n) (i.e., n = 2, 4, 6, or 8) as it is a convenient parameter that completely characterizes each.

For each technology scenario (i.e., n value) seven curves are plotted. The single solid (——) curve portrays the performance of the "OPT" model, in which no mechanisms for handling the cache-consistency problem (and, therefore, no overheads) are incorporated. Thus the throughput of the "OPT" model, as is demonstrated in the figure, provides a ceiling for all the other models. The "Store-Controller" model.s results are depicted by the two dash-and-dot (_._) curves (S1) and (S2). Curve (S1), which is always the dominating curve, is for the case where there is no data sharing between the caches, and (S2) is when 50 % data sharing is introduced. And finally, there are two curves for each of the two implementations of the "CC/PTB" approach. The two dashed (---) curves (P1) and (P2) belong to the "CC/PTB/P"

FACILITY	AVERAGE UTILIZATION	NUMBER ENTRIES	AVERAGE TIME/TRAN	SEIZING TRANS. NO.	PREEMPTING TRANS. NO.
GBUS	.474	1199	3.959		
LBUS1	.999	6514	1.535	59	
LBUS2	.702	1702	4.126	72	
DRP11	.396	678	5.846	37	
DRP12	.405	695	5.833		
DRP13	.411	703	5.849		
DRP14	.369	629	5.879		
DRP15	.398	681	5.856		
KRP1	.479	1199	4.000		
KRP2	.485	1214	4.000		
RSP2	.287	719	3.997	52	
DRP21	.198	496	4.000		

Figure (9)

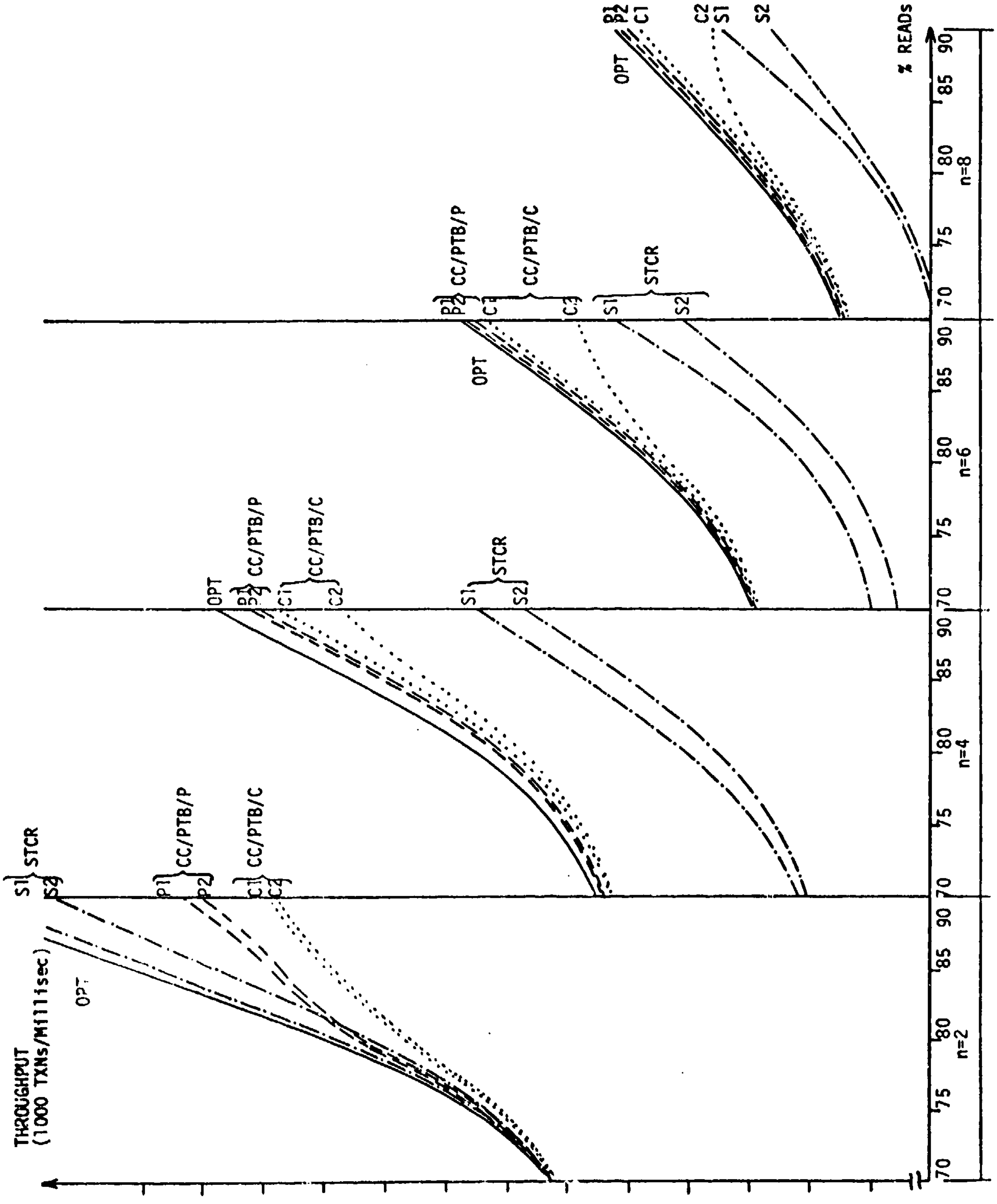

THROUGHPUT (1000 TXNs/Millisec)
% READs
n=2
n=4
n=6
n=8
OPT
STCR
CC/PTB/P
CC/PTB/C
S1
S2
P1
P2
C1
C2

Figure (10)

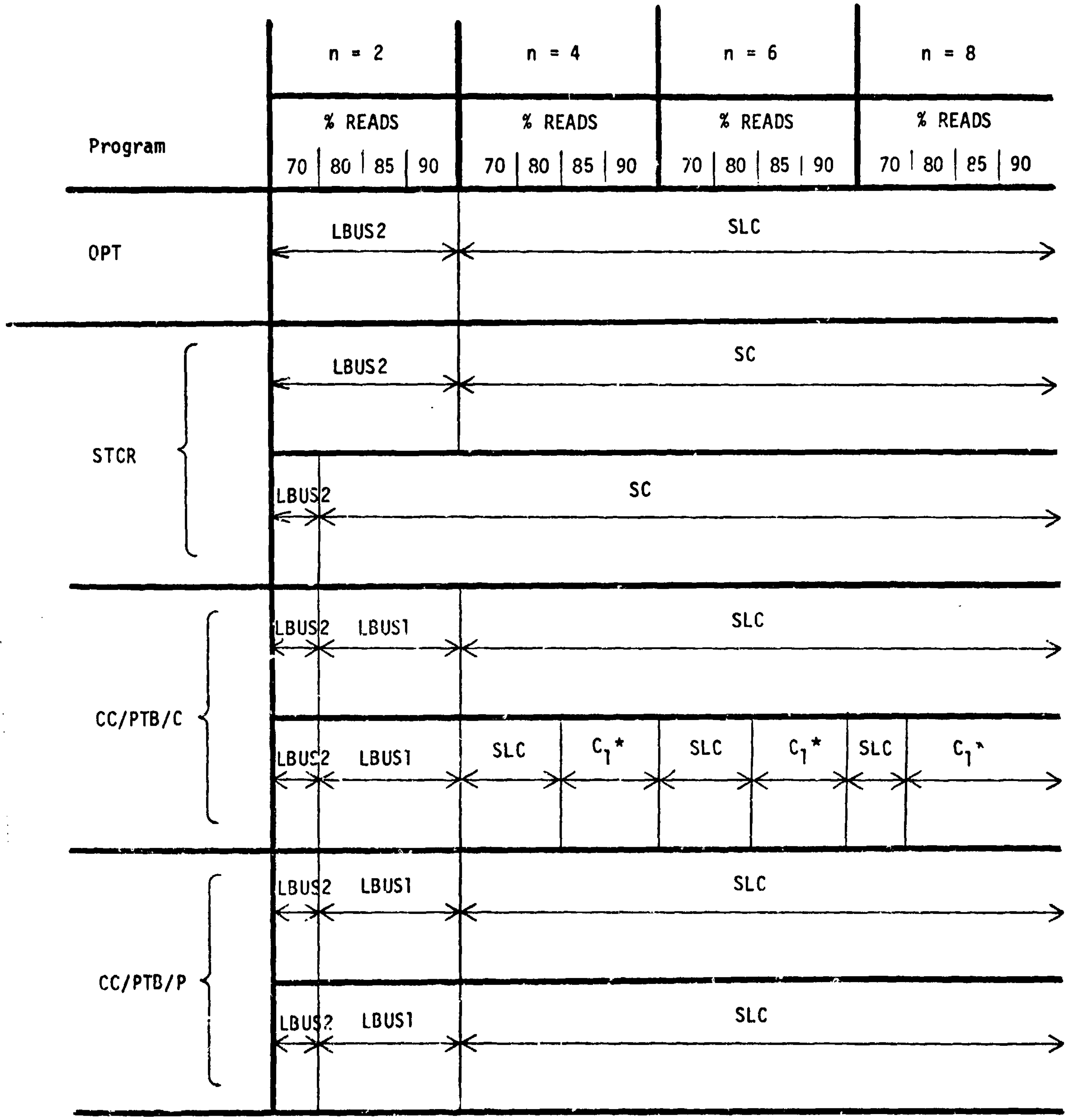
implementation, while the two dotted (...) ones (C1) and (C2) are for "CC/PTB/C." The two dominating curves in both implementations, namely the (P1) and (C1) curves, are for the case where the load is uniformly distributed among all cache-modules. The two other curves (P2) and (C2) depict the performances under the linearly distributed load.

As Figure 10 demonstrates, the "CC/PTB" architecture, in both its implementations, completely dominates the "Store-Controller" in three out of the four technology scenarios. Only in the first scenario (n = 2) does the "Store-Controller" show a performance advantage and only for high READ rates. The remaining part of this section will be devoted to an analysis of the different factors affecting the performance patterns demonstrated in Figure 10. Of particular help in conducting this analysis is the information of Figure 11 depicting the system bottlenecks in all the cases tested.

There are two basic patterns that deserve separate analysis. The distinctive pattern of n = 2, and the pattern common among the three other scenarios, n = 4, 6, and 8. To analyze the latter we will arbitrarily pick the case of n = 6 as our analysis vehicle.

V.5.1. Case of n = 2

To many, the most surprising aspect of these results will be the almost identical shapes of the "OPT" curve, and the "Store-Controller" curve (S1). To understand why this is so, we first note from Figure 11 that in both models LBUS2 (i.e., the local bus of level 2) is the bottleneck. Now, the difference between the "Store-Controller" model



*C_1 is the Cache-Module carrying 30 percent of the load

Figure (11)

and the "OPT" model lies in the overhead necessary to handle the cache-consistency problem. All this overhead (in the "Store-Controller" model) is in the form of additional operations which all take place at the cache level. Thus, while removing this overhead (in the "OPT" model) will necessarily decrease the load on the cache level, it will not decrease the load on level 2, and in particular on LBUS2. Thus, since LBUS2 is the bottleneck in both cases, and since the load on it (by an "average" transaction) remains the same, the performances in both are very similar. For the same reasons, the other five models, namely, (S2), (P1), (P2), (C1), and (C2), have performances close to that of "OPT" for % READs ,= 80 (i.e., they all have LBUS2 as the system bottleneck).

The fact that LBUS2 is the bottleneck is itself, by the way, an interesting finding. With a hit ratio of 0.90 for READ requests and 1.00 for WRITE requests most of the "action" is clearly done at the cache level. It would, therefore, seem that LBUS1 must be saturated before LBUS2. The answer goes back to the parameter values of section V.2. Notice that the transaction size for level 2 (64 bytes) is eight times larger than that for level 1 (8 bytes). This means that a single transmission on LBUS2 will be approximately eight times longer in time than a single transmission on LBUS1. Thus, although the absolute number of transmissions on LBUS2 is less than that on LBUS1, the utilization of LBUS2 in this case is higher.

Another interesting observation relates to curve (S2) of the "Store Controller." Curve (S2) is always below curve (S1) because in (S2) the "Store Controller" (SC) is the system bottleneck with a

saturation point lower than that of LBUS2. The reason this happens is that the high degree of data sharing between the caches (and which is "orchestrated" by the Store-Controller) means a higher utilization of the Store-Controller by the "average" READ/WRITE request. Notice also that the two curves (S1) and (S2) diverge as the % of READs increases. This is merely a reflection of the pattern by which the READ and WRITE operations utilize the two respective bottlenecks, LBUS2 and SC. (Straightforward analytic calculations would demonstrate that the effective capacity of LBUS2 increases faster than does that of SC as the % of READs increases.)

Next let us turn our attention to the "CC/PTB" results. Notice first the deflections in curves (P1), (P2), (C1), and (C2). This happens because at approximately 80 % READs LBUS1 (and not LBUS2) becomes the system bottleneck in the four cases. However, even though LBUS1 is the bottleneck for both "CC/PTB/P" and "CC/PTB/C", "CC/PTB/P" clearly dominates. This simply is because an "average" READ/WRITE request utilizes LBUS1 less often in "CC/PTB/P" than it does in "CC/PTB/C." For example, in CC/PTB/P when a requested data item is found by the processor not to be in the cache level, a request is sent to main memory via LBUS1. In CC/PTB/C, on the other hand, a CPU request must first go to a cache-module (through LBUS1), only to be found unavailable, and then forwarded by the cache-module to main memory through LBUS1 again.

Notice finally the negligible effect that the load distribution on the cache-modules has on performance (i.e., curves (C1) and (C2) are similar, as well as curves (P1) and (P2)). The reasons for this are

completely analogous to the ones explaining the close resemblance between the "OPT" curve and curve (S1). In other words, changing the load distribution does not affect the utilization of LBUS1. As long as no new bottleneck develops because of the change in the load distribution, LBUS1 will remain to be the bottleneck, and thus maintain approximately the same throughput. The utilization of the cache-module that carries the highest load under the linear load distribution for both "CC/PTB/P" and "CC/PTB/C" turns out never to exceed 60 %, which is far below the 100 % mark that has to be approached before it would replace LBUS1 as the system bottleneck. This, in a sense, is very comforting to know. It shows that the behavior of the models is, to a large extent, insensitive to the shape of the load distribution on the cache-modules that we used.

V.5.2. Case of n = 6

Most of the ideas of the above discussion are applicable to the case of n=6. For example, "OPT," "CC/PTB/P".s (P1) and (P2), and "CC/PTB/C".s (C1) all have almost identical performances because they all have the same bottleneck, namely, the Storage Level Controller (SLC).

Notice, on the other hand, that because the bus is now relatively faster in comparison to the other system components, and in particular to the Store-Controller (SC), LBUS2 ceases to be the bottleneck in the two "Store-Controller" models. Instead, SC is now the bottleneck, and the degradation in performance is quite evident. However, notice that even though SC is the bottleneck for both (S1) and (S2), the

throughputs are quite different. The reason for this is that an "average" READ/WRITE request in (S2) (where there is 50 % data sharing) requires more "services" from the "Store-Controller" than does an "average" READ/WRITE request in (S1). (When data sharing exists between the cache-modules, the SC has, for example, to invalidate redundant copies when a data item is modified.)

The only remaining result that deserves some explanation, is the deflection exhibited in curve (C2) of "CC/PTB/C" at % READs = 80. The reason for this behavior is that somewhere between % READs = 80 and % READs = 85 the most heavily loaded cache-module (i.e., the one carrying 30 % of the load) replaces SLC as the system.s bottleneck. (See Figure 11). Notice that this does not happen in "CC/PTB/P.s" curve (P2) even though the same linear load distribution is used. The reason for this is because, even though, the cache-modules in both cases are subjected to the same load distribution, they are not subjected to the same load. This, of course, is because the READ/WRITE operations in the "CC/PTB/P" implementation use the cache-modules less often.

V.6. Conclusion

The above results clearly indicate that no one approach dominates over all four technology scenarios. Technology, therefore, must remain as an element of some uncertainty.

It is important to realize, though, that what is really important in our technology forecasts is not the absolute values of the different speeds, but rather the relative values for the different hardware components. And the most important such value is the relative speed of the bus vis-a-vis the processing components that use it. Our results clearly demonstrate that the faster the bus is relative to everything else, the more appealling the "CC/PTB" approach becomes. More specifically, when the bus is four times as fast as the cache or faster (i.e., $n \geq 4$), the "CC/PTB" approach provides a 20 to 60 % performance advantage over the "Store-Controller" approach.

But, what perhaps is the most interesting finding, is the fact that for three out of the four technology scenarios (with $n \geq 4$) the performance of our "CC/PTB" scheme is very close to that of "OPT."

Notice that in the above statements no attempt was made to single out any of the two different implementation schemes of the "CC/PTB" approach. One of the interesting findings in the simulation results is that the performances of both schemes are very close indeed. Our own intuition was that the "CC/PTB/P" implementation would display a performance advantage. It was clear, that by utilizing "fast" processors that would overlap address translation with arbitrating for the bus, the utilizations of the bus and the cache-modules would decrease. Although the utilizations were indeed lower, this did not materialize into the higher performance we anticipated. The reason: the execution of the INFOPLEX storage hierarchy operations and storage mechanisms was such that the storage level controller (SLC), whose utilization is independent of the "CC/PTB" scheme used, would, in most

cases, be the first component to saturate. The only exception to this is when, under the linear load distribution case, the most heavily utilized cache-module developed into the system.s bottleneck. In such a case the higher performance potential of the "CC/PTB/P" scheme was indeed realized.

VI. CONCLUSION

This paper is a report on an ongoing research effort at the M.I.T. Sloan School of Management to study the multicache-consistency problem in highly parallel multi-processor computer systems.

There are three basic approaches proposed in the literature to handle the multicache-consistency problem: The "Broadcasting" approach, the "Store-Controller" approach, and the "Multics" approach. However, serious drawbacks could be identified in each. A new approach called the "CC/PTB" was, therefore, developed. It attempts to minimize performance degradation by minimizing the overhead of maintaining cache-consistency.

The "CC/PTB" approach was implemented in the INFOPLEX storage hierarchy and evaluated using simulation modeling. The results are very favorable.

This work opens up many areas for further investigation. No mention was made in this paper, for example, of the replacement algorithms at the cache level. It would be interesting to find out the value of implementing a "sophisticated" algorithm such as the "Least Recently Used" (LRU) algorithm (or versions of it) as opposed to a naive algorithm e.g., random replacement that requires much less hardware overhead.

We have assumed, as is common in the literature, that all WRITE operations for a particular data item are preceded by READ operations. Relaxing this constraint, and developing efficient algorithms to exploit both this relaxation and the architecture of the storage hierarchy is definately worth investigating.

And finally, the "CC/PTB" architecture should be exploited in the development of algorithms that would improve the reliability of the data storage hierarchy. The automatic data repair algorithms, for example, are particularly interesting and promising.

BIBLIOGRAPHY

1. Censier, Lucien M. and Feautrier, Paul. "A New Solution to Coherence Problems in Multicache Systems," IEEE Trans. on Computers, Vol. C-27, No. 12, (Dec., 1978), 1112-1118.

2. Ferrari, D. Computer Systems Performance Evaluation. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1978.

3. Goodrich II, E. R. "A Distributed Operating System for a Central Shared Bus Multiprocessor System." Unpublished Master Thesis, M.I.T., 1980.

4. Greenberg, B. An internal Honeywell memorandum, 1978.

5. Kaplan, K. R. and Winder, R. O. "Cache-based Computer Systems," Computer, (March, 1973), 31-36.

6. Lam, C. Y. "Simulation Studies of the DSH-II Data Storage Hierarchy System." M.I.T. Sloan School Internal Report No. M010-7906-04, 1979a.

7. Lam, C. Y. "Data Storage Hierarchy Systems for Data Base Computers." Unpublished Ph.D. Thesis, M.I.T., 1979b.

8. Madnick, S. E. "INFOPLEX - Hierarchical Decomposition of a Large Information Management System Using a Microprocessor Complex," AFIPS Proc., Vol. 44, 1975, 581-587.

9. Madnick, S. E. "The INFOPLEX Database Computer: Concepts and Directions," Proc. IEEE Computer Conference, Feb. 26, 1979, 168-176.

10. Matick, R. Computer Storage Systems & Technology. New York: John Wiley & Sons, 1977.

11. Tang, C. K. "Cache System Design in the Tightly Coupled Multiprocessor System," AFIPS Proc., Vol. 45, 1976, 746-753.

12. Toong, H. D.; Strommen, S. O.; and Goodrich II, E. R. "A General Multi-Microprocessor Interconnection Mechanism for Non-Numeric Processing," Proc. of the Fifth Workshop on Computer Architecture for Non-Numeric Processing, 1980, 115-123.

APPENDIX (I) : The "OPT" Program

```
//TAR1 JOB TAR,
// PROFILE='DEFER',
// TIME=5
//*PASSWORD SCUBA
//GPSS PROC
//C   EXEC  PGM=DAG01,TIME=&TLIMIT
//STEPLIB DD DSN=POTLUCK.LIBRARY.GPSS.LOAD,DISP=SHR
//DOUTPUT DD SYSOUT=PROFILE=PRINT,DCB=BLKSIZE=931
//DINTERO DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=1880
//DSYMTAB DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7112
//DREPTGEN DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//DINTWORK DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=2680
// PEND
//STEP1  EXEC  GPSS,PARM=C,TLIMIT=9
//DINPUT1  DD *
       REALLOCATE FUN,5,QUE,10,FAC,50,BVR,200,BLO,2000,VAR,50
       REALLOCATE FSV,50,HSV,10,COM,40000

*****************************  OPT


*************************
*                       *
* TXN PARM USAGE        *
*                       *
*  P1   CPU ID          *
*  P2   TXN ARRIVAL TIME*
*  P3   TXN COMPL TIME  *
*  P4   TXN EXEC TIME   *
*  P11  DUMMY           *
*                       *
*************************

*************************
*                       *
*  MODEL COMPONENTS     *
*                       *
* BUSES: GBUS, LBUS1,...*
* CACHES: D11,...D15    *
* LEVEL CONTRL: K1,K2   *
* REQ PROCS: R2         *
* DEVICES: D21          *
* STORAGE : RI, RO      *
* STORAGE : SI, SO      *
* STORAGE : TI, TO      *
* STORAGE : AI, AC      *
* STORAGE : OI, OO      *
*                       *
*************************

*************************
*                       *
* MODEL PARAMETERS      *
*                       *
*************************
```

```
      INITIAL    X$MODEL,102      MODEL OPT
      INITIAL    X$MAXMP,10       DEGREE OF MULTIPROG PER CPU
      INITIAL    X$MREAD,700      % READ REQ
      INITIAL    X$PIN1,900       CONDITIONAL PROB OF FINDING DATA IN L1
      INITIAL    X$PIN2,1000      IN L2
      INITIAL    X$DEK1,2         DEVICE SERVICE TIME IN L1
     INITIAL    X$DEK2,4          IN L2
      INITIAL    X$BEK1,1         BUS SERVICE TIME FOR 8 BYTE BLOCK
      INITIAL    X$BEK8,8         BUS SERVICE TIME FOR 64 BYTE BLOCK
      INITIAL    X$CBUS,1         BUS TIME BETWEEN CPU AND CACHE
      INITIAL    X$PEK,4          DIRECTORY LOOK UP
      INITIAL    X$CEK,4          CONTROLLER SERV TIME
      INITIAL    X$REPL,8         TIME TO USE REPL ALG & STORE IN L1
      INITIAL    X$PDEX1,6        LOOKUP PLUS READ TIME OF CACHE
      INITIAL    X$TIMER,10000    SIMULATION TIME

*************************
*                       *
* SAVEVALUES            *
*                       *
* NTXN TOTAL TXN PROC.  *
* SUMX TOTAL EXEC TIMES *
* SUMW TOTAL WAIT TIMES *
* SUMT TOTAL ELAPSED TIM*
*                       *
*************************

*************************
*                       *
*  VARIABLES            *
*                       *
*************************

MRESP FVARIABLE  (X$SUMT/X$NTXN)       MEAN RESP TIME
TXNT  VARIABLE   P3-P2                 TXN ELAPSED TIME
TXNW  VARIABLE   P3-P2-P4              TXN WAIT TIME
TXNX  VARIABLE   P4                    TXN EXEC TIME

*************************
*                       *
*  TABLES               *
*                       *
*************************

TXNT  TABLE      V$TXNT,100,100,100
TXNW  TABLE      V$TXNW,100,100,100
TXNX  TABLE      V$TXNX,100,100,100

*************************
*                       *
*  FUNCTIONS            *
*                       *
*************************
```

```
 WICHW FUNCTION   P1,D5
1,WWW11/2,WWW12/3,WWW13/4,WWW14/5,WWW15

 WICHA FUNCTION   P1,D5
1,AAA11/2,AAA12/3,AAA13/4,AAA14/5,AAA15

***********************
*                     *
* STORAGE FOR L(1)    *
* CACHES              *
*                     *
***********************

       STORAGE    S$RID11,10/S$SID11,10/S$TID11,10/S$AID11,10
       STORAGE    S$RID12,10/S$SID12,10/S$TID12,10/S$AID12,10
       STORAGE    S$RID13,10/S$SID13,10/S$TID13,10/S$AID13,10
       STORAGE    S$RID14,10/S$SID14,10/S$TI[illegible],10/S$AID14,10
       STORAGE    S$RID15,10/S$SID15,10/S$TID15,10/S$AID15,10

***********************
*                     *
* STORAGE FOR DEVICES *
*                     *
***********************

       STORAGE    S$RID21,10/S$SID21,10/S$TID21,10
***********************
*                     *
* STORAGE FOR REQ PROC *
*                     *
***********************

       STORAGE    S$RIR2,10/S$SIR2,10/S$TIR2,10/S$AIR2,10/S$OIR2,10

***********************
*                     *
*  STORAGE FOR K1     *
*                     *
***********************

       STORAGE    S$ROK1,10/S$SOK1,10/S$TIK1,10/S$AIK1,10/S$OOK1,10

***********************
*                     *
* STORAGE FOR K2,K3,K4 *
*                     *
***********************

       STORAGE    S$RIK2,10/S$SIK2,10/S$TIK2,10/S$AIK2,10/S$OIK2,10
       STORAGE    S$ROK2,10/S$SOK2,10/S$TOK2,10/S$ADK2,10/S$OOK2,10

***********************
*                     *
* BOOLEAN VARIABLES   *
*                     *
```

```
************************
*                      *
* BV FOR READ-THROUGH  *
*                      *
************************

RTOK2 BVARIABLE  FNU$GBUS*SNF$TIK1

************************
*                      *
* BV FOR L 1           *
*                      *
************************

DKR1  BVARIABLE  FNU$LBUS1*SNF$ROK1
DKS1  BVARIABLE  FNU$LBUS1*SNF$SOK1
DKO1  BVARIABLE  FNU$LBUS1*SNF$OOK1
KDT11 BVARIABLE  FNU$LBUS1*SNF$TID11
KDT12 BVARIABLE  FNU$LBUS1*SNF$TID12
KDT13 BVARIABLE  FNU$LBUS1*SNF$TID13
KDT14 BVARIABLE  FNU$LBUS1*SNF$TID14
KDT15 BVARIABLE  FNU$LBUS1*SNF$TID15
KDA11 BVARIABLE  FNU$LBUS1*SNF$AID11
KDA12 BVARIABLE  FNU$LBUS1*SNF$AID12
KDA13 BVARIABLE  FNU$LBUS1*SNF$AID13
KDA14 BVARIABLE  FNU$LBUS1*SNF$AID14
KDA15 BVARIABLE  FNU$LBUS1*SNF$AID15

************************
*                      *
* BV FOR INTER LEVEL COM*
*                      *
************************

KKR12 BVARIABLE  FNU$GBUS*SNF$RIK2
KKS12 BVARIABLE  FNU$GBUS*SNF$SIK2
KKO12 BVARIABLE  FNU$GBUS*SNF$SOK2
KKT21 BVARIABLE  FNU$GBUS*SNF$TIK1
KKA21 BVARIABLE  FNU$GBUS*SNF$AIK1

************************
*                      *
* BV FOR L(2) OPS      *
*                      *
************************

KRR2  BVARIABLE  FNU$LBUS2*SNF$RIR2
KRS2  BVARIABLE  FNU$LBUS2*SNF$SIR2
KRT2  BVARIABLE  FNU$LBUS2*SNF$TIR2
KRA2  BVARIABLE  FNU$LBUS2*SNF$AIR2
KRO2  BVARIABLE  FNU$LBUS2*SNF$OIR2
ROR21 BVARIABLE  FNU$LBUS2*SNF$RID21
ROS21 BVARIABLE  FNU$LBUS2*SNF$SID21
```

```
RDT21  BVARIABLE   FNU$LBUS2*SNF$TID21
DKS2   BVARIABLE   FNU$LBUS2*SNF$SCK2
DKT2   BVARIABLE   FNU$LBUS2*SNF$TCK2
DKA2   BVARIABLE   FNU$LBUS2*SNF$AOK2
RKR2   BVARIABLE   FNU$LBUS2*SNF$ROK2
RKO2   BVARIABLE   FNU$LBUS2*SNF$OOK2
RKA2   BVARIABLE   FNU$LBUS2*SNF$AOK2

*************************
*                       *
*    MACROS             *
*                       *
*************************

*************************
*                       *
* MACRO -USE            *
* #A   FACILITY         *
*
* #B   USAGE TIME

 USE   STARTMACRO
       SEIZE       #A
       ADVANCE     #B
       ASSIGN      4+,#B
       RELEASE     #A
       ENDMACRO

*************************
*                       *
*  MACRO - SEND         *
*                       *
* #A  FROM              *
* #B  TO                *
* #C  VIA               *
* #D  TRANSIT TIME      *
* #E  BV FOR SEND OP  * 
*                       *
*************************

 SEND  STARTMACRO
       TEST E      #E,1
       ENTER       #B
       SEIZE       #C
       ADVANCE     #D
       ASSIGN      4+,#D
       RELEASE     #C
       LEAVE       #A
       ENDMACRO

*************************
*                       *
*  MACRO - FINI         *
*                       *
```

```
*************************

FINI  STARTMACRO
      MARK          3
      SAVEVALUE     NTXN+,1
      SAVEVALUE     SUMX+,V$STXNX
      SAVEVALUE     SUMW+,V$STXNW
      SAVEVALUE     SUMT+,V$STXNT
      SAVEVALUE     MRESP,V$MRESP
      ASSIGN        1,0
      ASSIGN        2,0
      ASSIGN        3,0
      ASSIGN        4,0
      ENDMACRO

*-----------------------------------------------------------------*
*                                                                 *
*        BEGIN SIMULATION                                         *
*                                                                 *
*-----------------------------------------------------------------*

      SIMULATE
*************************
*                       *
*   CPU #1              *
*                       *
*************************

      RMULT         3,5,7,9,11,13,15,17

CPU1  GENERATE      ,,,X$VAXMP,,,F
STAR1 PRIORITY      9                 SET HIGH P FOR NEW TXN
      MARK          2                 ARRIVAL TIME
      ASSIGN        1,1               CPU ID
      ADVANCE       X$CBUS
      TRANSFER      .X$READ,WWW1,RRR1
RRR1  TRANSFER      .X$PIN1,NIN11,RIN11

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

RIN11 ENTER         RID11             PUT TXN IN READ REQ BUFFER
USE   MACRO         DRP11,X$RDEX1     SEARCH AND READ CACHE
      LEAVE         RID11             FREE BUFFER
      ADVANCE       X$CBUS
FINI  MACRO
      TRANSFER      ,STAR1            A NEW TXN

*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
```

```
*************************

NIN11 ENTER     RID11              PUT IN READ REQ BUFFER
USE   MACRO     DRP11,X$REX        SEARCH DIRECTORY
      PRIORITY  0                  RESET PRIORITY
SEND  MACRO     RID11,ROK1,LBUS1,X$BEX1,BV$DKR1

      TRANSFER  ,COMR              TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW1  ENTER     SID11              PUT TXN IN WRITE REQ BUFFER

USE   MACRO     DRP11,X$RDEX1       WRITE DATA IN CACHE

      PRIORITY  0                  RESET TXN PRIORITY
SEND  MACRO     SID11,SOK1,LBUS1,X$BEX8,BV$DKS1

      SPLIT     1,COMW

FINI  MACRO

      TRANSFER  ,STAR1             A NEW TXN

*************************
*                       *
*   CPU #2              *
*                       *
*************************

CPU2  GENERATE  ,,,X$MAXMP,,,F
STAR2 PRIORITY  9                  SET HIGH P FOR NEW TXN
      MARK      2                  ARRIVAL TIME
      ASSIGN    1,2                CPU ID
      ADVANCE   X$CBUS
      TRANSFER  .X$NREAD,WWW2,RRR2
RRR2  TRANSFER  .X$PIN1,NIN12,RIN12

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

RIN12 ENTER     RID12              PUT TXN IN READ REQ BUFFER
USE   MACRO     DRP12,X$RDEX1      SEARCH AND READ CACHE
      LEAVE     RID12              FREE BUFFER
      ADVANCE   X$CBUS
FINI  MACRO
      TRANSFER  ,STAR2             A NEW TXN
```

```
*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
*************************

NIN12 ENTER       RID12              PUT IN READ REQ BUFFER
USE   MACRO       DRP12,X$REX        SEARCH DIRECTORY
      PRIORITY    0                  RESET PRIORITY
SEND  MACRO       RID12,ROK1,LBUS1,X$BEX1,BV$DKR1

      TRANSFER    ,COMR              TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW2  ENTER       SID12              PUT TXN IN WRITE REQ BUFFER

USE   MACRO       DRP12,X$RDEX1       WRITE DATA IN CACHE

      PRIORITY    0                  RESET TXN PRIORITY
SEND  MACRO       SID12,SOK1,LBUS1,X$BEX8,BV$DKS1

      SPLIT       1,COMW

FINI  MACRO

      TRANSFER    ,STAR2             A NEW TXN

*************************
*                       *
*   CPU #3              *
*                       *
*************************

CPU3  GENERATE    ,,,X$WAXMP,,,F
STAR3 PRIORITY    9                  SET HIGH P FOR NEW TXN
      MARK        2                  ARRIVAL TIME
      ASSIGN      1,3                CPU ID
      ADVANCE     X$CBUS
      TRANSFER    ,X$NREAD,WWW3,RRR3
RRR3  TRANSFER    ,X$PIN1,NIN13,RIN13

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

RIN13 ENTER       RID13              PUT TXN IN READ REQ BUFFER
USE   MACRO       DRP13,X$RDEX1      SEARCH AND READ CACHE
      LEAVE       RID13              FREE BUFFER
```

```
       ADVANCE    X$CBUS
 FINI  MACRO
       TRANSFER   ,STAR3             A NEW TXN

************************
*                      *
* DATA IS NOT IN CACHE *
*                      *
************************

 NIN13 ENTER      RID13              PUT IN READ REQ BUFFER
 USE   MACRO      DRP13,X$REX        SEARCH DIRECTORY
       PRIORITY   0                  RESET PRIORITY
 SEND  MACRO      RID13,ROK1,LBUS1,X$BEX1,BV$DKR1

       TRANSFER   ,COMR              TO COMMON CODE FOR READ

************************
*                      *
* WRITE REQUEST TO CACHE*
*                      *
************************

 WWW3  ENTER      SID13              PUT TXN IN WRITE REQ BUFFER

 USE   MACRO      DRF13,X$RDEX1       WRITE DATA IN CACHE

       PRIORITY   0                  RESET TXN PRIORITY

 SEND  MACRO      SID13,SOK1,LBUS1,X$BEX8,BV$DKS1

       SPLIT      1,COMW

 FINI  MACRO

       TRANSFER   ,STAR3

************************
*                      *
*   CPU 44             *
*                      *
************************

 CPU4  GENERATE   ,,,X$MAXMP,,,F
 STAR4 PRIORITY   9                  SET HIGH P FOR NEW TXN
       MARK       2                  ARRIVAL TIME
       ASSIGN     1,4                CPU ID
       ADVANCE    X$CBUS
       TRANSFER   .X$NREAD,WWW4,RRR4
 RRR4  TRANSFER   .X$PIN1,NIN14,RIN14

************************
*                      *
* DATA IS IN DATA CACHE *
*                      *
```

```
*************************

RIN14 ENTER       RID14                 PUT TXN IN READ REQ BUFFER
USE   MACRO       DRP14,X$RDEX1         SEARCH AND READ CACHE
      LEAVE       RID14                 FREE BUFFER
      ADVANCE     X$CBUS
FINI  MACRO
      TRANSFER    ,STAR4                A NEW TXN

*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
*************************

NIN14 ENTER       RID14                 PUT IN READ REQ BUFFER
USE   MACRO       DRP14,X$REX           SEARCH DIRECTORY
      PRIORITY    0                     RESET PRIORITY
SEND  MACRO       RID14,ROK1,LBUS1,X$BEX1,BV$DKR1

      TRANSFER    ,COMR                 TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW4  ENTER       SID14                 PUT TXN IN WRITE REQ BUFFER

USE   MACRO       DRP14,X$RDEX1          WRITE DATA IN CACHE

      PRIORITY    0                     RESET TXN PRIORITY

SEND  MACRO       SID14,SOK1,LBUS1,X$BEX8,BV$DKS1

      SPLIT       1,COMW

FINI  MACRO

      TRANSFER    ,STAR4                A NEW TXN

*************************
*                       *
*   CPU #5              *
*                       *
*************************

CPU5  GENERATE    ,,,XSMAXMP,,,F
STAR5 PRIORITY    9                     SET HIGH P FOR NEW TXN
      MARK        2                     ARRIVAL TIME
      ASSIGN      1,5                   CPU ID
      ADVANCE     X$CBUS
      TRANSFER    .X$NREAD,WWW5,RRR5
RRR5  TRANSFER    .X$PIN1,NIN15,RIN15
```

```
************************
*                      *
* DATA IS IN DATA CACHE *
*                      *
************************

 RIN15 ENTER      RID15            PUT TXN IN READ REQ BUFFER
 USE   MACRO      DRP15,X$RDEX1    SEARCH AND READ CACHE
       LEAVE      RID15            FREE BUFFER
       ADVANCE    X$CBUS
 FINI  MACRO
       TRANSFER   ,STAR5           A NEW TXN

************************
*                      *
* DATA IS NOT IN CACHE *
*                      *
************************

 NIN15 ENTER      RID15            PUT TXN IN READ REQ BUFFER
 USE   MACRO      DRP15,X$REX      SEARCH DIRECTORY
       PRIORITY   0                RESET PRIORITY
 SEND  MACRO      RID15,ROK1,LBUS1,X$BEX1,BV$DKR1

       TRANSFER   ,COMR            TO COMMON CODE FOR READ

************************
*                      *
* WRITE REQUEST TO CACHE*
*                      *
************************

 WWW5  ENTER      SID15            PUT TXN IN WRITE REQ BUFFER

 USE   MACRO      DRP15,X$RDEX1     WRITE DATA IN CACHE

       PRIORITY   0                RESET TXN PRIORITY

 SEND  MACRO      SID15,SOK1,LBUS1,X$BEX8,BV$DKS1

       SPLIT      1,COMW

 FINI  MACRO

       TRANSFER   ,STAR5           A NEW TXN

*--------------------------------------------------------------*
*                                                              *
*   COMMON CODE FOR READ REQUEST                               *
*                                                              *
*--------------------------------------------------------------*

 COMR  ASSIGN     11,0
```

```
USE   MACRO     KRP1,X$KEX
SEND  MACRO     ROK1,RIK2,GBUS,X$BEX1,BV$KKR12
USE   MACRO     KRP2,X$KEX
SEND  MACRO     RIK2,RIR2,LBUS2,X$BEX1,BV$KRR2
USE   MACRO     RRP2,X$REX

*---------------------------------------------------------*
*                                                         *
*  READ DATA IS FOUND IN L(2)                             *
*                                                         *
*---------------------------------------------------------*

************************
*                      *
* DATA IS IN D21       *
*                      *
************************
RRR21 ASSIGN    11,0
SEND  MACRO     RIR2,RID21,LBUS2,X$BEX1,BV$RDR21
USE   MACRO     DRP21,X$DEX2
SEND  MACRO     RID21,TOK2,LBUS2,X$BEX8,BV$DKT2

************************
*                      *
* READ-THROUGH TO L(1) *
*                      *
************************

USE   MACRO     KRP2,X$KEX
SEND  MACRO     TOK2,TIK1,GBUS,X$BEX8,BV$RTOK2
*---------------------------------------------------------*
*                                                         *
* STORE DATA INTO L(1) AS RESULT OF A READ-THROUGH        *
*                                                         *
*---------------------------------------------------------*
STOR1 ASSIGN    11,0
USE   MACRO     KRP1,X$KEX
```

```
       SPLIT        1,FNSWICHW
       TERMINATE

*************************
*                       *
* RT STORE INTO D11     *
*                       *
*************************

WWW11  ASSIGN       11,0

SEND   MACRO        TIK1,TID11,LBUS1,X$BEX8,BV$KDT11

USE    MACRO        DRP11,X$RPLR

       TRANSFER     .X$POV1,MOV11,OVL11
NOV11  LEAVE        TID11
       ADVANCE      X$CBUS

FINI   MACRO

       TRANSFER     ,STAR1

OVL11  SPLIT        1,OVF11
       ADVANCE      X$CBUS

FINI   MACRO

       TRANSFER     ,STAR1
OVF11  ASSIGN       11,0

SEND   MACRO        TID11,OOK1,LBUS1,X$BEX1,BV$DKO1

       TRANSFER     ,OVL1

*************************
*                       *
* RT STORE INTO D12     *
*                       *
*************************

WWW12  ASSIGN       11,0

SEND   MACRO        TIK1,TID12,LBUS1,X$BEX8,BV$KDT12

USE    MACRO        DRP12,X$RPLR

       TRANSFER     .X$POV1,NOV12,OVL12
NOV12  LEAVE        TID12
       ADVANCE      X$CBUS

FINI   MACRO

       TRANSFER     ,STAR2
```

```
 OVL12 SPLIT        1,OVF12
       ADVANCE      X$CBUS

 FINI  MACRO

       TRANSFER     ,STAR2
 OVF12 ASSIGN       11,0

 SEND  MACRO        TID12,OOK1,LBUS1,X$BEX1,3,$DKD1

       TRANSFER     ,OVL1

*************************
*                       *
* RT STORE INTO D13     *
*                       *
*************************

 BWW13 ASSIGN       11,0

 SEND  MACRO        TIK1,TID13,LBUS1,X$BEX8,3,$KDT13

 USE   MACRO        DRP13,X$RPLR

       TRANSFER     .X$POV1,NOV13,OVL13
 NOV13 LEAVE        TID13
       ADVANCE      X$CBUS

 FINI  MACRO

       TRANSFER     ,STAR3

 OVL13 SPLIT        1,OVF13
       ADVANCE      X$CBUS

 FINI  MACRO

       TRANSFER     ,STAR3
 OVF13 ASSIGN       11,0

 SEND  MACRO        TID13,OOK1,LBUS1,X$BEX1,3,$DKD1

       TRANSFER     ,OVL1

*************************
*                       *
* RT STORE INTO D14     *
*                       *
*************************

 BWW14 ASSIGN       11,0

 SEND  MACRO        TIK1,TID14,LBUS1,X$BEX8,3,$KDT14

 USE   MACRO        DRP14,X$RPLR
```

```
       TRANSFER    .X$FOV1,NOV14,OVL14
 NOV14 LEAVE       TID14
       ADVANCE     X$CBUS

 FINI  MACRO

       TRANSFER    ,STAR4

 OVL14 SPLIT       ,OVF14
       ADVA        X$CBUS

 FINI  MACRO

       TRANSFER    ,STAR4
 OVF14 ASSIGN      11,0

 SEND  MACRO       TID14,OOK1,LBUS1,X$BEX1,BV$DKO1

       TRANSFER    ,OVL1

*************************
*                       *
* RT STORE INTO D15     *
*                       *
*************************

 WWW15 ASSIGN      11,0

 SEND  MACRO       TIK1,TID15,LBUS1,X$BEX8,BV$KDT15

 USE   MACRO       DRP15,X$RPLR

       TRANSFER    .X$POV1,NOV15,OVL15
 NOV15 LEAVE       TID15
       ADVANCE     X$CBUS

 FINI  MACRO

       TRANSFER    ,STAR5

 OVL15 SPLIT       1,OVF15
       ADVANCE     X$CBUS

 FINI  MACRO

       TRANSFER    ,STAR5
 OVF15 ASSIGN      11,0

 SEND  MACRO       TID15,OOK1,LBUS1,X$BEX1,BV$DKO1

       TRANSFER    ,OVL1

*************************
*                       *
```

```
* HANDLE OVF FROM L(1)  *
*                       *
*************************

OVL1  ASSIGN     11,0
USE   MACRO      KRP1,X$KEX
SEND  MACRO      COK1,SIK2,GBUS,X$BEX1,BV$KKO12
USE   MACRO      KRP2,X$KEX
SEND  MACRO      OIK2,SIR2,LBUS2,X$BEX1,BV$KRO2
USE   MACRO      RRP2,X$REX
      LEAVE      OIR2
      TERMINATE

*---------------------------------------------------------*
*                                                         *
*  COMMON CODE FOR STORE-BEHIND                           *
*                                                         *
*---------------------------------------------------------*

COMM  ASSIGN     11,0
USE   MACRO      KRP1,X$KEX
SEND  MACRO      SOK1,SIK2,GBUS,X$BEX8,BV$KKS12
USE   MACRO      KRP2,X$KEX
SEND  MACRO      SIK2,SIR2,LBUS2,X$BEX8,BV$KRS2
USE   MACRO      RRP2,X$REX

SEND  MACRO      SIR2,SID21,LBUS2,X$BEX8,BV$RDS21
USE   MACRO      DRP21,X$DEX2
SEND  MACRO      SID21,AOK2,LBUS2,X$BEX1,BV$DKS2

*---------------------------------------------------------*
*                                                         *
*  ACK FROM L(2) TO L(1)                                  *
*                                                         *
*---------------------------------------------------------*

ACK21 ASSIGN     11,0
USE   MACRO      KRP2,X$KEX
```

```
 SEND  MACRO      AOK2,AIK1,GBUS,X$BEX1,BV$KKA21

 USE   MACRO      KRP1,X$KEX

       SPLIT      1,FN$WICHA
       TERMINATE

*************************
*                       *
* ACK HANDLED BY D11    *
*                       *
*************************

 AAA11 ASSIGN     11,0

 SEND  MACRO      AIK1,AID11,LBUS1,X$BEX1,BV$KDA11

 USE   MACRO      DRP1',X$REX

       LEAVE      AID11
       TERMINATE

*************************
*                       *
* ACK HANDLED BY D12    *
*                       *
*************************

 AAA12 ASSIGN     .1,0

 SEND  MACRO      AIK1,AID12,LBUS1,X$BEX1,BV$KDA12

 USE   MACRO      DRP12,X$REX

       LEAVE      AID12
       TERMINATE

*************************
*                       *
* ACK HANDLED BY D13    *
*                       *
*************************

 AAA13 ASSIGN     11.0

 SEND  MACRO      AIK1,AID13,LBUS1,X$BEX1,BV$KDA13

 USE   MACRO      DRP13,X$REX

       LEAVE      AID13
       TERMINATE

*************************
*                       *
* ACK HANDLED BY D14    *
```

```
*                         *
***************************

 AAA14 ASSIGN     11.0

 SEND  MACRO      AIK1,AID14,LBLS1,XSBEX1,BV$KDA14

 USE   MACRO      DRP14,X$REX

       LEAVE      AID14
       TERMINATE

***************************
*                         *
* ACK HANDLED BY D15      *
*                         *
***************************

 AAA15 ASSIGN     11.0

 SEND  MACRO      AIK1,AID15,LBLS1,X$BEX1,BV$KDA15

 USE   MACRO      DRP15,X$REX

       LEAVE      AID15
       TERMINATE

*------------------------------------------------------------*
*                                                            *
*  SIMULATION CONTROL                                        *
*                                                            *
*------------------------------------------------------------*

        GENERATE   X$TIMER
        TERMINATE  1
        START      1
        END
$$ END
```

APPENDIX (II): The "STCR" Program

```
//TAR1 JOB TAR,
// PROFILE='DEFER',
// TIME=5
//*PASSWORD SCUBA
//GPSS PROC
//C    EXEC  PGM=DAG01,TIME=&TLIMIT
//STEPLIB DD DSN=POTLUCK.LIBRARY.GPSS.LOAD,DISP=SHR
//DOUTPUT DD SYSOUT=PROFILE=PRINT,DCB=BLKSIZE=931
//DINTERO DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=1880
//DSYMTAB DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7112
//DREPTGEN DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//DINTWORK DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=2680
// PEND
//STEP1  EXEC  GPSS,PARM=C,TLIMIT=9
//DINPUT1  DD *
       REALLOCATE FUN,5,QUE,10,FAC,50,BVR,200,BLO,2000,VAR,50
       REALLOCATE FSV,50,HSV,10,COM,40000

******************************  STCR
*************************
*                       *
* TXN PARM USAGE        *
*                       *
*  P1   CPU ID          *
*  P2   TXN ARRIVAL TIME*
*  P3   TXN COMPL TIME  *
*  P4   TXN EXEC TIME   *
*  P11  DUMMY           *
*                       *
*************************

*************************
*                       *
*  MODEL COMPONENTS     *
*                       *
* BUSES: GBUS, LBUS1,.. *
* CACHES: D11,...D15    *
* LEVEL CONTRL: K1,K2   *
* REQ PROCS: R1,R2      *
* DEVICES: D21          *
* STORAGE : RI, RO      *
* STORAGE : SI, SO      *
* STORAGE : TI, TO      *
* STORAGE : AI, AO      *
* STORAGE : DI, DO      *
*                       *
*************************

*************************
*                       *
* MODEL PARAMETERS      *
*                       *
*************************

       INITIAL    X$MODEL,       MODEL STCR
```

```
      INITIAL     X$MAXMP,10      DEGREE OF MULTIPROG PER CPU
      INITIAL     X$NREAD,900     % READ REQ
      INITIAL     X$PIN1,900      CONDITIONAL PROB OF FINDING DATA IN L1
      INITIAL     X$PIN2,1000     IN L2
      INITIAL     X$DEX1,2        DEVICE SERVICE TIME IN L1
      INITIAL     X$DEX2,4        IN L2
      INITIAL     X$BEX1,1        BUS SERVICE TIME FOR 8 BYTE BLOCK
      INITIAL     X$BEX8,8        BUS SERVICE TIME FOR 64 BYTE BLOCK
      INITIAL     X$CBUS,1        BUS SERVICE TIME BETWEEN CPU & CACHE
      INITIAL     X$REX,4         DIRECTORY LOOK UP TIME
      INITIAL     X$KEX,4         CONTROLLER SERV TIME
      INITIAL     X$RPLR,8        TIME TO USE RPL ALG AND STORE IN L1
      INITIAL     X$RDEX1,6       LOOKUP PLUS READ TIME OF CACHE
      INITIAL     X$TIMER,10000   SIMULATION TIME
      INITIAL     X$PRV1,1000     WHEN READING (&NOT IN CACHE):
*                                      % OF TIME ANOTHER CACHE DOES NOT
*                                      CONTAIN THE LINE AS A PRIVATE LINE
      INITIAL     X$PRV2,1000     WHEN WRITING A LINE (& IN CACHE):
*                                      PROB. LINE WAS NOT JUST DECLARED
*                                      PRIVATE BY ANOTHER CACHE
      INITIAL     X$SHR,1000      WHEN WRITING (&LINE IN CACHE & LINE
*                                      NOT JUST DECLARED PRIVATE BY
*                                      ANOTHER CACHE):
*                                      PROB THAT ONE OR MORE
*                                      OTHER CACHES DON'T SHARE THE LINE
      INITIAL     X$CNT,1000      PROB THAT 'STORE-BEHIND
*                                      ACK COUNTER' = 0
      INITIAL     X$SCS,4         STORE CONTROLLER SEARCHES DIREC
      INITIAL     X$RDEX,8        WRITE INTO CACHE+UPDT DIREC
      INITIAL     X$SCU,8         STCR UPDATES ITS DIREC
      INITIAL     X$SCSU,8        STCR SEARCHES & UPDATES ITS DIREC
      INITIAL     X$RDEX2,2       CACHE RECEIVES A CHANGE OF
*                                      STATUS ACK ... & SO IT WRITES
      INITIAL     X$REX1,8        CACHE UPDATES DIREC
      INITIAL     XSDECR,4        CACHE DECR 'STORE-BEHIND ACK CNT'

*************************
*                       *
* SAVEVALUES            *
*                       *
* NTXN TOTAL TXN PROC.  *
* SUMX TOTAL EXEC TIMES *
* SUMY TOTAL WAIT TIMES *
* SUMT TOTAL ELAPSED TIM*
*                       *
*************************

*************************
*                       *
*  VARIABLES            *
*                       *
*************************

 MRESP FVARIABLE  (X$SUMT/X$NTXN)       MEAN RESP TIME
 TXNT  VARIABLE   P3-P2                 TXN ELAPSED TIME
```

```
 TXNW  VARIABLE   P3-P2-P4                 TXN WAIT TIME
 TXNX  VARIABLE   P4                       TXN EXEC TIME

************************
*                      *
*  TABLES              *
*                      *
************************

 TXNT  TABLE      V$TXNT,100,100,100
 TXNW  TABLE      V$TXNW,100,100,100
 TXNX  TABLE      V$TXNX,100,100,100

************************
*                      *
*  FUNCTIONS           *
*                      *
************************

 WICHW FUNCTION   P1,D5
1,WWW11/2,WWW12/3,WWW13/4,WWW14/5,WWW15

 WICHA FUNCTION   P1,D5
1,AAA11/2,AAA12/3,AAA13/4,AAA14/5,AAA15

************************
*                      *
* STORAGE FOR L(1)     *
* CACHES               *
*                      *
************************

      STORAGE    S$RID11,10/S$SID11,10/S$TID11,10/S$AID11,10
      STORAGE    S$RID12,10/S$SID12,10/S$TID12,10/S$AID12,10
      STORAGE    S$RID13,10/S$SID13,10/S$TID13,10/S$AID13,10
      STORAGE    S$RID14,10/S$SID14,10/S$TID14,10/S$AID14,10
      STORAGE    S$RID15,10/S$SID15,10/S$TID15,10/S$AID15,10

************************
*                      *
* STORAGE FOR DEVICES  *
*                      *
************************

      STORAGE    S$RID21,10/S$SID21,10/S$TID21,10
************************
*                      *
* STORAGE FOR REQ PROC *
*                      *
************************

      STORAGE    S$RIR1,10/S$SIR1,10/S$TIR1,10/S$AIR1,10/S$DIR1,10
      STORAGE    S$RIR2,10/S$SIR2,10/S$TIR2,10/S$AIR2,10/S$DIR2,10

************************
```

```
*                       *
*  STORAGE FOR K1       *
*                       *
*************************

       STORAGE     S$ROK1,10/S$SOK1,10/S$TIK1,10/S$AIK1,10/S$OOK1,10

*************************
*                       *
* STORAGE FOR K2,K3,K4  *
*                       *
*************************

       STORAGE     S$RIK2,10/S$SIK2,10/S$TIK2,10/S$AIK2,10/S$OIK2,10
       STORAGE     S$ROK2,10/S$SOK2,10/S$TOK2,10/S$AOK2,10/S$OOK2,10

*************************
*                       *
* BOOLEAN VARIABLES     *
*                       *
*************************

*************************
*                       *
* BV FOR READ-THROUGH   *
*                       *
*************************

 DRKO1 BVARIABLE   FNU$LBUS1*SNF$OIR1*SNF$OOK1
 RTOK2 BVARIABLE   FNU$GBUS*SNF$TIK1

*************************
*                       *
* BV FOR L(1)           *
*                       *
*************************

 DRR1   BVARIABLE   FNU$LBUS1*SNF$RIR1
 DRS1   BVARIABLE   FNU$LBUS1*SNF$SIR1
 DRA1   BVARIABLE   FNU$LBUS1*SNF$AIR1
 DRO1   BVARIABLE   FNU$LBUS1*SNF$OIR1
 RR1    BVARIABLE   SNF$RIR1
 SR1    BVARIABLE   SNF$SIR1
 RKR1   BVARIABLE   FNU$LBUS1*SNF$RCK1
 RDS11  BVARIABLE   FNU$LBUS1*SNF$SID11
 RDS12  BVARIABLE   FNU$LBUS1*SNF$SID12
 RDS13  BVARIABLE   FNU$LBUS1*SNF$SID13
 RDS14  BVARIABLE   FNU$LBUS1*SNF$SID14
 RDS15  BVARIABLE   FNU$LBUS1*SNF$SID15
 D1D2   BVARIABLE   FNU$LBUS1*SNF$SID11*SNF$SID12
 D2D3   BVARIABLE   FNU$LBUS1*SNF$SID12*SNF$SID13
 D3D4   BVARIABLE   FNU$LBUS1*SNF$SID13*SNF$SID14
 D4D5   BVARIABLE   FNU$LBUS1*SNF$SID14*SNF$SID15
 D5D1   BVARIABLE   FNU$LBUS1*SNF$SID15*SNF$SID11
 DKR1   BVARIABLE   FNU$LBUS1*SNF$ROK1
```

```
 DKS1   BVARIABLE   FNU$LBUS1*SNF$SSOK1
 DKO1   BVARIABLE   FNU$LBUS1*SNF$SOOK1
 KDT11  BVARIABLE   FNU$LBUS1*SNF$STID11
 KDT12  BVARIABLE   FNU$LBUS1*SNF$STID12
 KDT13  BVARIABLE   FNU$LBUS1*SNF$STID13
 KDT14  BVARIABLE   FNU$LBUS1*SNF$STID14
 KDT15  BVARIABLE   FNU$LBUS1*SNF$STID15
 KDA11  BVARIABLE   FNU$LBUS1*SNF$SAID11
 KDA12  BVARIABLE   FNU$LBUS1*SNF$SAID12
 KDA13  BVARIABLE   FNU$LBUS1*SNF$SAID13
 KDA14  BVARIABLE   FNU$LBUS1*SNF$SAID14
 KDA15  BVARIABLE   FNU$LBUS1*SNF$SAID15

************************
*                      *
* BV FOR INTER LEVEL COV*
*                      *
************************

 KKR12  BVARIABLE   FNU$GBUS*SNF$SRIK2
 KKS12  BVARIABLE   FNU$GBUS*SNF$SSIK2
 KKO12  BVARIABLE   FNU$GBUS*SNF$SSOK2
 KKT21  BVARIABLE   FNU$GBUS*SNF$STIK1
 KKA21  BVARIABLE   FNU$GBUS*SNF$SAIK1

************************
*                      *
* BV FOR L(2) GPS      *
*                      *
************************

 KRR2   BVARIABLE   FNU$LBUS2*SNF$SRIR2
 KRS2   BVARIABLE   FNU$LBUS2*SNF$SSIR2
 KRT2   BVARIABLE   FNU$LBUS2*SNF$STIR2
 KRA2   BVARIABLE   FNU$LBUS2*SNF$SAIR2
 KRO2   BVARIABLE   FNU$LBUS2*SNF$SOIR2
 RDR21  BVARIABLE   FNU$LBUS2*SNF$SRID21
 RDS21  BVARIABLE   FNU$LBUS2*SNF$SSID21
 RDT21  BVARIABLE   FNU$LBUS2*SNF$STID21
 DKS2   BVARIABLE   FNU$LBUS2*SNF$SSOK2
 DKT2   BVARIABLE   FNU$LBUS2*SNF$STOK2
 DKA2   BVARIABLE   FNU$LBUS2*SNF$SAOK2
 RKR2   BVARIABLE   FNU$LBUS2*SNF$SROK2
 RKO2   BVARIABLE   FNU$LBUS2*SNF$SOOK2
 RKA2   BVARIABLE   FNU$LBUS2*SNF$SAOK2

************************
*                      *
*    MACROS            *
*                      *
************************

************************
*                      *
```

```
* MACRO -USE              *
* #A  FACILITY           *
*
* #B  USAGE TIME

 USE   STARTMACRO
       SEIZE       #A
       ADVANCE     #B
       ASSIGN      4+,#B
       RELEASE     #A
       ENDMACRO

*************************
*                       *
*  MACRO - SEND         *
*                       *
* #A  FROM              *
* #B  TO                *
* #C  VIA               *
* #D  TRANSIT TIME      *
* #E  BV FOR SEND OP    *
*                       *
*************************

 SEND  STARTMACRO
       TEST E      #E,1
       ENTER       #B
       SEIZE       #C
       ADVANCE     #D
       ASSIGN      4+,#D
       RELEASE     #C
       LEAVE       #A
       ENDMACRO

*************************
*                       *
*  MACRO - FINI         *
*                       *
*************************

 FINI  STARTMACRO
       MARK        3
       SAVEVALUE   NTXN+,1
       SAVEVALUE   SUMX+,V$TXNX
       SAVEVALUE   SUMW+,V$TXNW
       SAVEVALUE   SUMT+,V$TXNT
       SAVEVALUE   MRESP,V$MRESP
       ASSIGN      1,0
       ASSIGN      2,0
       ASSIGN      3,0
       ASSIGN      4,0
       ENDMACRO

**-----------------------------------------------------------*
*                                                           *
```

```
*          BEGIN SIMULATION                                        *
*                                                                  *
*------------------------------------------------------------------*

       SIMULATE
*************************
*                       *
*   CPU #1              *
*                       *
*************************

       RMULT      3.5,7.9,11,13,15,17

 CPU1  GENERATE   ,,,XSMAXMP,,,F
 STAR1 PRIORITY   9                  SET HIGH P FOR NEW TXN
       MARK       2                  ARRIVAL TIME
       ASSIGN     1,1                CPU ID
       ADVANCE    XSCBUS
       TRANSFER   .XSNREAD,WWW1,RRR1
 RRE1  TRANSFER   .XSPIN1,NIN11,RIN11

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

 RIN11 ENTER      RID11              PUT TXN IN READ REQ BUFFER
 USE   MACRO      DRP11,XSRDEX1      SEARCH AND READ CACHE
       LEAVE      RID11              FREE BUFFER
       ADVANCE    XSCBUS
 FINI  MACRO
       TRANSFER   ,STAR1             A NEW TXN

*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
*************************

 NIN11 ENTER      RID11              PUT IN READ REQ BUFFER
 USE   MACRO      DRP11,XSREX        SEARCH DIRECTORY
       PRIORITY   0                  RESET PRIORITY
 SEND  MACRO      RID11,RIR1,LBUS1,XSBEX1,BVSDRR1

       TRANSFER   ,COMR              TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

 WWW1  ENTER      SID11              PUT TXN IN WRITE REQ BUFFER
```

```
USE     MACRO        DRP11,X$REX      WRITE DATA IN CACHE

        PRIORITY     0                 RESET TXN PRIORITY
SEND    MACRO        SID11,SIR1,LBUS1,X$BEX1,BV$DRS1


TRY1    ASSIGN       11,0
        TRANSFER     .X$PRV2,,NTP11
USE     MACRO        RRP1,X$SCS
        LEAVE        SIR1
        BUFFER
        TEST E       SR1,1
        ENTER        SIR1
        TRANSFER     ,TRY1
NTP11   ASSIGN       11,0
USE     MACRO        RRP1,X$SCSU
        TRANSFER     .X$SHR,SHR11,SHR01
SHR01   ASSIGN       11,0
SEND    MACRO        SIR1,SID11,LBUS1,X$BEX1,BV$RDS11
USE     MACRO        DRP11,X$RDEX2
SEND    MACRO        SID11,SOK1,LBUS1,X$BEX8,BV$DKS1
        SPLIT        1,COMW
FIN1    MACRO
        TRANSFER     ,STAR1
SHR11   ASSIGN       11,0
        TEST E       BV$DID2,1
        ENTER        SID11
        ENTER        SIC
        SEIZE        LBUS1
        ADVANCE      X$BEX1
        ASSIGN       4+,X$BEX
        RELEASE      LBUS
        LEAVE        SIR1
        SPLIT        1,SHR21
USE     MACRO        DRP11,X$RDEX2
SEND    MACRO        SID11,SOK1,LBUS1,X$BEX8,BV$DKS1
        SPLIT        1,COMW
FIN1    MACRO
        TRANSFER     ,STAR1
SHR21   ASSIGN       11,0
USE     MACRO        DRP12,X$REX1
        LEAVE        SID12
        TERMINATE
*************************
*                       *
*   CPU #2              *
*                       *
*************************

CPU2    GENERATE     ,,,X$MAXMP,,,F
STAR2   PRIORITY     9                 SET HIGH P FOR NEW TXN
        MARK         2                 ARRIVAL TIME
        ASSIGN       1,2               CPU ID
        ADVANCE      X$CBUS
        TRANSFER     .X$NREAD,WWW2,RRR2
```

```
RRR2  TRANSFER    .XSPIN1,NIN12,RIN12

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

RIN12 ENTER       RID12              PUT TXN IN READ REQ BUFFER
USE   MACRO       DRP12,X$RDEX1      SEARCH AND READ CACHE
      LEAVE       RID12              FREE BUFFER
      ADVANCE     X$CBUS
FINI  MACRO
      TRANSFER    ,STAR2             A NEW TXN

*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
*************************

NIN12 ENTER       RID12              PUT IN READ REQ BUFFER
USE   MACRO       DRP12,X$REX        SEARCH DIRECTORY
      PRIORITY    0                  RESET PRIORITY
SEND  MACRO       RID12,RIR1,LBUS1,X$SEX1,BV$DRR1

      TRANSFER    ,COMR              TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW2  ENTER       SID12              PUT TXN IN WRITE REQ BUFFER

USE   MACRO       DRP12,X$REX        WRITE DATA IN CACHE

      PRIORITY    0                  RESET TXN PRIORITY
SEND  MACRO       SID12,SIR1,LBUS1,X$SEX1,BV$DRS1


TRY2  ASSIGN      11,0
      TRANSFER    .X$PRV2,,NTP12
USE   MACRO       RRP1,X$SCS
      LEAVE       SIR1
      BUFFER
      TEST E      SR1,1
      ENTER       SIR1
      TRANSFER    ,TRY2
NTP12 ASSIGN      11,0
USE   MACRO       RRP1,X$SCSU
      TRANSFER    .X$SHR,SHR12,SHR02
SHR02 ASSIGN      11,0
SEND  MACRO       SIR1,SID12,LBUS1,X$SEX1,BV$RDS12
```

```
 USE   MACRO       DRP12,X$RDEX2
 SEND  MACRO       SID12,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT       1,COMW
 FINI  MACRO
       TRANSFER    ,STAR2
 SHR12 ASSIGN      11,0
       TEST E      BV$D203,1
       ENTER       SID12
       ENTER       SID13
       SEIZE       LBUS1
       ADVANCE     X$BEX1
       ASSIGN      4+,X$BEX1
       RELEASE     LBUS1
       LEAVE       SIR1
       SPLIT       1,SHR22
 USE   MACRO       DRP12,X$RDEX2
 SEND  MACRO       SID12,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT       1,COMW
 FINI  MACRO
       TRANSFER    ,STAR2
 SHR22 ASSIGN      11,0
 USE   MACRO       DRP13,X$REX1
       LEAVE       SID13
       TERMINATE
************************
*                      *
*   CPU #3             *
*                      *
************************

 CPU3  GENERATE    ,,,X$MAXMP,,,F
 STAR3 PRIORITY    9                SET HIGH P FOR NEW TXN
       MARK        2                ARRIVAL TIME
       ASSIGN      1,3              CPU ID
       ADVANCE     X$CBUS
       TRANSFER    .X$NREAD,WWW3,RRR3
 RRR3  TRANSFER    .X$PIN1,NIN13,RIN13

************************
*                      *
* DATA IS IN DATA CACHE *
*                      *
************************

 RIN13 ENTER       RID13            PUT TXN IN READ REQ BUFFER
 USE   MACRO       DRP13,X$RDEX1    SEARCH AND READ CACHE
       LEAVE       RID13            FREE BUFFER
       ADVANCE     X$CBUS
 FINI  MACRO
       TRANSFER    ,STAR3           A NEW TXN

************************
*                      *
* DATA IS NOT IN CACHE *
*                      *
```

```
*************************

NIN13 ENTER      RID13            PUT IN READ REQ BUFFER
USE   MACRO      DRP13,XSREX      SEARCH DIRECTORY
      PRIORITY   0                RESET PRIORITY
SEND  MACRO      RID13,RIR1,LBUS1,XSBEX1,BVSDRR1

      TRANSFER   ,CCMR            TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW3  ENTER      SID13            PUT TXN IN WRITE REQ BUFFER

USE   MACRO      DRP13,XSREX      WRITE DATA IN CACHE

      PRIORITY   0                RESET TXN PRIORITY

SEND  MACRO      SID13,SIR1,LBUS1,X$BEX1,BV$DRS1


TRY3  ASSIGN     11,0
      TRANSFER   .XSPRV2,,NTP13
USE   MACRO      RRP1,X$SCS
      LEAVE      SIR1
      BUFFER
      TEST E     SR1,1
      ENTER      SIR1
      TRANSFER   ,TRY3
NTP13 ASSIGN     11,0
USE   MACRO      RRP1,X$SCSU
      TRANSFER   .X$SHR,SHR13,SHR03
SHR03 ASSIGN     11,0
SEND  MACRO      SIR1,SID13,LBUS1,X$BEX1,BV$RDS13
USE   MACRO      DRP13,X$RDEX2
SEND  MACRO      SID13,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT      1,CCW*
FINI  MACRO
      TRANSFER   ,STAR3
SHR13 ASSIGN     11,0
      TEST E     BVSD3D4,1
      ENTER      SID13
      ENTER      SID14
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      SIR1
      SPLIT      1,SHR23
USE   MACRO      DRP13,XSRDEX2
SEND  MACRO      SID13,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT      1,COW*
```

```
 FINI   MACRO
        TRANSFER     ,STAR3
 SHR23  ASSIGN       11,0
 USE    MACRO        DRP14,X$REX1
        LEAVE        SID14
        TERMINATE
************************
*                      *
*   CPU #4             *
*                      *
************************

 CPU4   GENERATE     ,,,X$MAXMP,,,F
 STAR4  PRIORITY     9                 SET HIGH P FOR NEW TXN
        MARK         2                 ARRIVAL TIME
        ASSIGN       1,4               CPU ID
        ADVANCE      X$CBUS
        TRANSFER     .X$NREAD,WWW4,RRR4
 RRR4   TRANSFER     .X$PIN1,NIN14,RIN14

************************
*                      *
* DATA IS IN DATA CACHE *
*                      *
************************

 RIN14  ENTER        RID14             PUT TXN IN READ REQ BUFFER
 USE    MACRO        DRP14,X$RDEX1     SEARCH AND READ CACHE
        LEAVE        RID14             FREE BUFFER
        ADVANCE      X$CBUS
 FINI   MACRO
        TRANSFER     ,STAR4            A NEW TXN

************************
*                      *
* DATA IS NOT IN CACHE  *
*                      *
************************

 NIN14  ENTER        RID14             PUT IN READ REQ BUFFER
 USE    MACRO        DRP14,X$REX       SEARCH DIRECTORY
        PRIORITY     0                 RESET PRIORITY
 SEND   MACRO        RID14,RIR1,LBUS1,X$BEX1,BV$DRR1

        TRANSFER     ,COMR             TO COMMON CODE FOR READ

************************
*                      *
* WRITE REQUEST TO CACHE*
*                      *
************************

 WWW4   ENTER        SID14             PUT TXN IN WRITE REQ BUFFER

 USE    MACRO        DRP14,X$REX      WRITE DATA IN CACHE
```

```
       PRIORITY   0               RESET TXN PRIORITY
SEND   MACRO      SID14,SIR1,LBUS1,X$BEX1,BV$DRS1

TRY4   ASSIGN     11,0
       TRANSFER   .X$PRV2,,NTP14
USE    MACRO      RRP1,X$SCS
       LEAVE      SIR1
       BUFFER
       TEST E     SR1,1
       ENTER      SIR1
       TRANSFER   ,TRY4
NTP14  ASSIGN     11,0
USE    MACRO      RRP1,X$SCSU
       TRANSFER   .X$SHR,SHR14,SHR04
SHR04  ASSIGN     11,0
SEND   MACRO      SIR1,SID14,LBUS1,X$BEX1,BV$RDS14
USE    MACRO      DRP14,X$RDEX2
SEND   MACRO      SID14,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT      1,COMW
FINI   MACRO
       TRANSFER   ,STAR4
SHR14  ASSIGN     11,0
       TEST E     BV$D4DS,1
       ENTER      SID14
       ENTER      SID15
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4+,X$BEX1
       RELEASE    LBUS1
       LEAVE      SIR1
       SPLIT      1,SHR24
USE    MACRO      DRP14,X$RDEX2
SEND   MACRO      SID14,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT      1,COMW
FINI   MACRO
       TRANSFER   ,STAR4
SHR24  ASSIGN     11,0
USE    MACRO      DRP15,X$REX1
       LEAVE      SID15
       TERMINATE
*************************
*                       *
*   CPU #5              *
*                       *
*************************

CPU5   GENERATE   ,,,X$MAXMP,,,F
STAR5  PRIORITY   9               SET HIGH P FOR NEW TXN
       MARK       2               ARRIVAL TIME
       ASSIGN     1,5             CPU ID
       ADVANCE    X$CBUS
       TRANSFER   .X$NREAD,WWW5,RRR5
```

```
RRR5  TRANSFER    .X$PIN1,NIN15,RIN15

*************************
*                       *
* DATA IS IN DATA CACHE *
*                       *
*************************

RIN15 ENTER       RID15              PUT TXN IN READ REQ BUFFER
USE   MACRO       DRP15,X$RDEX1      SEARCH AND READ CACHE
      LEAVE       RID15              FREE BUFFER
      ADVANCE     X$CBUS
FINI  MACRO
      TRANSFER    ,STAR5             A NEW TXN

*************************
*                       *
* DATA IS NOT IN CACHE  *
*                       *
*************************

NIN15 ENTER       RID15              PUT IN READ REQ BUFFER
USE   MACRO       DRP15,X$REX        SEARCH DIRECTORY
      PRIORITY    0                  RESET PRIORITY
SEND  MACRO       RID15,RIR1,LBUS1,X$BEX1,BV$DRR1

      TRANSFER    ,COMR              TO COMMON CODE FOR READ

*************************
*                       *
* WRITE REQUEST TO CACHE*
*                       *
*************************

WWW5  ENTER       SID15              PUT TXN IN WRITE REQ BUFFER

USE   MACRO       DRP15,X$REX        WRITE DATA IN CACHE

      PRIORITY    0                  RESET TXN PRIORITY

SEND  MACRO       SID15,SIR1,LBUS1,X$BEX1,BV$DRS1

TRY5  ASSIGN      11,0
      TRANSFER    .X$PRV2,,NTP15
USE   MACRO       RRP1,X$SCS
      LEAVE       SIR1
      BUFFER
      TEST E      SR1,1
      ENTER       SIR1
      TRANSFER    ,TRY5
NTP15 ASSIGN      11,0
USE   MACRO       RRP1,X$SCSU
      TRANSFER    .X$SHR,SHR15,SHR05
SHR05 ASSIGN      11,0
```

```
SEND  MACRO       SIR1,SID15,LBUS1,X$BEX1,BV$RDS15
USE   MACRO       DRP15,X$RDEX2
SEND  MACRO       SID15,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COM*
FINI  MACRO
      TRANSFER    ,STAR5
SHR15 ASSIGN      11,0
      TEST E      BV$CDSD1,1
      ENTER       SID15
      ENTER       SID11
      SEIZE       LBUS1
      ADVANCE     X$BEX1
      ASSIGN      4+,X$BEX1
      RELEASE     LBUS1
      LEAVE       SIR1
      SPLIT       1,SHR25
USE   MACRO       DRP15,X$RDEX2
SEND  MACRO       SID15,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COM*
FINI  MACRO
      TRANSFER    ,STAR5
SHR25 ASSIGN      11,0
USE   MACRO       DRP11,X$REX1
      LEAVE       SID11
      TERMINATE
*-------------------------------------------------------------*
*                                                             *
*   COMMON CODE FOR READ REQUEST                              *
*                                                             *
*-------------------------------------------------------------*

COMR  ASSIGN      11,0
USE   MACRO       RRP1,X$SCS
      TRANSFER    ,X$PRV1,,NTPV
      LEAVE       RIR1
      BUFFER
      TEST E      BV$RR1,1
      ENTER       RIR1
      TRANSFER    ,COMR
NTPV  ASSIGN      11,0
SEND  MACRO       RIR1,ROK1,LBUS1,X$BEX1,BV$RKR1

USE   MACRO       KRP1,X$KEX

SEND  MACRO       ROK1,RIK2,GBUS,X$BEX1,BV$KKR12

USE   MACRO       KRP2,X$KEX

SEND  MACRO       RIK2,RIR2,LBUS2,X$BEX1,BV$KRR2

USE   MACRO       RRP2,X$REX

*-------------------------------------------------------------*
*                                                             *
```

```
*  READ DATA IS FOUND IN L(2)                                        *
*                                                                    *
*--------------------------------------------------------------------*


***************************
*                         *
* DATA IS IN D21          *
*                         *
***************************

 RRR21 ASSIGN     11,0
 SEND  MACRO      RIR2,RID21,LBUS2,X$BEX1,BV$RDR21
 USE   MACRO      DRP21,X$DEX2
 SEND  MACRO      RID21,TOK2,LBUS2,X$BEX8,BV$DKT2


***************************
*                         *
* READ-THROUGH TO L(1)    *
*                         *
***************************

 USE   MACRO      KRP2,X$KEX
 SEND  MACRO      TOK2,TIK1,GBUS,X$BEX8,BV$RTOK2
*--------------------------------------------------------------------*
*                                                                    *
* STORE DATA INTO L(1) AS RESULT OF A READ-THROUGH                   *
*                                                                    *
*--------------------------------------------------------------------*
 STOR1 ASSIGN     11,0
 USE   MACRO      KRP1,X$KEX
       SPLIT      1,FN$WICHW
       TERMINATE
***************************
*                         *
* RT STORE INTO D11       *
*                         *
***************************
 WWW11 ASSIGN     11,0
 SEND  MACRO      TIK1,TID11,LBUS1,X$BEX8,BV$KDT11
```

```
 USE   MACRO      DRP11,XSRPLR

       SPLIT      1,OVL11
       TEST E     BV$DRKD1,1
       ENTER      OIR1
       ENTER      OOK1
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4+,X$BEX1
       RELEASE    LBUS1
       LEAVE      TID11
       SPLIT      1,OVL1
 USE   MACRO      RRP1,X$SCU
       LEAVE      OIR1
       TERMINATE
OVL11  ASSIGN     11,0
       ADVANCE    X$CBUS
 FINI  MACRO
       TRANSFER   ,STAR1
*************************
*                       *
* RT STORE INTO D12     *
*                       *
*************************

WWW12 ASSIGN      11,0

 SEND  MACRO      TIK1,TID12,LBUS1,X$BEX8,BV$KDT12

 USE   MACRO      DRP12,X$RPLR

       SPLIT      1,OVL12
       TEST E     BV$DRKD1,1
       ENTER      OIR1
       ENTER      OOK1
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4+,X$BEX1
       RELEASE    LBUS1
       LEAVE      TID12
       SPLIT      1,OVL1
 USE   MACRO      RRP1,X$SCU
       LEAVE      OIR1
       TERMINATE
OVL12  ASSIGN     11,0
       ADVANCE    X$CBUS
 FINI  MACRO
       TRANSFER   ,STAR2
*************************
*                       *
* RT STORE INTO D13     *
*                       *
*************************
```

```
WWW13 ASSIGN     11,0

SEND  MACRO      TIK1,TID13,LBUS1,X$BEX8,BV$KDT13

USE   MACRO      DRP13,X$RPLR

      SPLIT      1,OVL13
      TEST E     BV$DRKO1,1
      ENTER      OIR1
      ENTER      OOK1
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      TID13
      SPLIT      1,OVL1
USE   MACRO      RRP1,X$SCU
      LEAVE      OIR1
      TERMINATE
OVL13 ASSIGN     11,0
      ADVANCE    X$CBUS
FINI  MACRO
      TRANSFER   ,STAR3
*****************************
*                           *
* RT STORE INTO D14         *
*                           *
*****************************

WWW14 ASSIGN     11,0

SEND  MACRO      TIK1,TID14,LBUS1,X$BEX8,BV$KDT14

USE   MACRO      DRP14,X$RPLR

      SPLIT      1,OVL14
      TEST E     BV$DRKO1,1
      ENTER      OIR1
      ENTER      OOK1
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      TID14
      SPLIT      1,OVL1
USE   MACRO      RRP1,X$SCU
      LEAVE      OIR1
      TERMINATE
OVL14 ASSIGN     11,0
      ADVANCE    X$CBUS
FINI  MACRO
      TRANSFER   ,STAR4
*****************************
```

```
*                      *
* RT STORE INTO D15    *
*                      *
************************

WWW15 ASSIGN     11,0

SEND  MACRO      TIK1,TID15,LBUS1,X$BEX8,BV$KDT15

USE   MACRO      DRP15,X$RP_R

      SPLIT      1,OVL15
      TEST E     BV$DRKD1,1
      ENTER      DIR1
      ENTER      DDK1
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      TID15
      SPLIT      1,OVL1
USE   MACRO      RRP1,X$SCU
      LEAVE      DIR1
      TERMINATE
OVL15 ASSIGN     11,0
      ADVANCE    X$CBUS
FINI  MACRO
      TRANSFER   ,STAR5
************************
*                      *
* HANDLE OVF FROM L(1) *
*                      *
************************

OVL1  ASSIGN     11,0

USE   MACRO      KRP1,X$KEX

SEND  MACRO      DDK1,DIK2,GBUS,X$BEX1,BV$KKD12

USE   MACRO      KRP2,X$KEX

SEND  MACRO      DIK2,DIR2,LBUS2,X$BEX1,BV$KRD2

USE   MACRO      RRP2,X$REX

      LEAVE      DIR2
      TERMINATE

*-------------------------------------------------------*
*                                                       *
*  COMMON CODE FOR STORE-BEHIND                         *
*                                                       *
```

```
*-----------------------------------------------------------*

 COMM  ASSIGN      11,0
 USE   MACRO       KRP1,X$KEX
 SEND  MACRO       SOK1,SIK2,GBUS,X$BEX8,BV$KKS12
 USE   MACRO       KRP2,X$KEX
 SEND  MACRO       SIK2,SIR2,LBUS2,X$BEX8,BV$KRS2
 USE   MACRO       RRP2,X$REX

 SEND  MACRO       SIR2,SID21,LBUS2,X$BEX8,BV$RDS21
 USE   MACRO       DRP21,X$DEX2
 SEND  MACRO       SID21,AOK2,LBUS2,X$BEX1,BV$DKS2

*-----------------------------------------------------------*
*                                                           *
*  AC' FROM L(2) TO L(1)                                    *
*                                                           *
*-----------------------------------------------------------*

 ACK21 ASSIGN      11,0
 USE   MACRO       KRP2,X$KEX
 SEND  MACRO       AOK2,AIK1,GBUS,X$BEX1,BV$KKA21
 USE   MACRO       KRP1,X$KEX
       SPLIT       1,FN$WICHA
       TERMINATE

************************
*                      *
* ACK HANDLED BY D11   *
*                      *
************************

 AAA11 ASSIGN      11,0
 SEND  MACRO       AIK1,AID11,LBUS1,X$BEX1,BV$KDA11
 USE   MACRO       DRP11,X$DECR
       TRANSFER    .X$CNT,,CNT11

       LEAVE       AID11
       TERMINATE
 CNT11 ASSIGN      11,0
 SEND  MACRO       AID11,AIR1,LBUS1,X$BEX1,BV$DRA1
```

```
USE   MACRO       RRP1,X$SCU
      LEAVE       AIR1

      TERMINATE
*************************
*                       *
* ACK HANDLED BY D12    *
*                       *
*************************

AAA12 ASSIGN      11,0

SEND  MACRO       AIK1,AID12,LBUS1,X$BEX1,BV$KDA12

USE   MACRO       DRP12,X$DECR
      TRANSFER    ,X$CNT,,CNT12

      LEAVE       AID12
      TERMINATE
CNT12 ASSIGN      11,0
SEND  MACRO       AID12,AIR1,LBUS1,X$BEX1,BV$DRA1
USE   MACRO       RRP1,X$SCU
      LEAVE       AIR1

      TERMINATE
*************************
*                       *
* ACK HANDLED BY D13    *
*                       *
*************************

AAA13 ASSIGN      11,0

SEND  MACRO       AIK1,AID13,LBUS1,X$BEX1,BV$KDA13

USE   MACRO       DRP13,X$DECR
      TRANSFER    ,X$CNT,,CNT13

      LEAVE       AID13
      TERMINATE
CNT13 ASSIGN      11,0
SEND  MACRO       AID13,AIR1,LBUS1,X$BEX1,BV$DRA1
USE   MACRO       RRP1,X$SCU
      LEAVE       AIR1

      TERMINATE
*************************
*                       *
* ACK HANDLED BY D14    *
*                       *
*************************

AAA14 ASSIGN      11,0

SEND  MACRO       AIK1,AID14,LBUS1,X$BEX1,BV$KDA14
```

```
 USE   MACRO      DRP14,X$DECR
       TRANSFER   .X$CNT,,CNT14

       LEAVE      AID14

       TERMINATE
 CNT14 ASSIGN     11,0
 SEND  MACRO      AID14,AIR1,LBUS1,X$BEX1,BV$DRA1
 USE   MACRO      RRP1,X$SCU
       LEAVE      AIR1
       TERMINATE
************************
*                      *
* ACK HANDLED BY D15   *
*                      *
************************

 AAA15 ASSIGN     11,0

 SEND  MACRO      A[illegible]1,AID15,LBUS1,X$BEX1,BV$KDA15

 USE   MACRO      DRP15,X$DECR
       TRANSFER   .X$CNT,,CNT15

       LEAVE      AID15
       TERMINATE
 CNT15 ASSIGN     11,0
 SEND  MACRO      AID15,AIR1,LBUS1,X$BEX1,BV$DRA1
 USE   MACRO      RRP1,X$SCU
       LEAVE      AIR1

       TERMINATE
*------------------------------------------------------------*
*                                                            *
*  SIMULATION CONTROL                                        *
*                                                            *
*------------------------------------------------------------*

       GENERATE   X$TIMER
       TERMINATE  1
       START      1
       END
$$ END
```

APPENDIX (III): The "CC/PTB/C" Program

```
FILE: CCC      VS1JOB   A4                CONVERSATIONAL MONITOR SYSTEM                PAGE 001

//TAR1 JOB TAR,
// PROFILE='DEFER',
// TIME=5
//*PASSWORD SCUBA
//GPSS PROC
//C   EXEC  PGM=DAG01,TIME=&TLIMIT
//STEPLIB DD DSN=POTLUCK.LIBRARY.GPSS.LOAD,DISP=SHR
//DOUTPUT DD SYSOUT=PROFILE=PRINT,DCB=BLKSIZE=931
//DINTERO DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=1380
//DSYMTAB DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7112
//DREPTGEN DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//DINTWORK DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=2680
// PEND
//STEP1  EXEC  GPSS.PARM=C,TLIMIT=9
//DINPUT1  DD *
       REALLOCATE FUN,6,QUE,10,FAC,50,BVR,200,BLO,2000,VAR,50
       REALLOCATE FSV,50,HSV,10,COM,39968

*****************************  CC/PTB/C
*************************
*                       *
* TXN PARM USAGE        *
*                       *
*  P1   CPU ID          *
*  P2   TXN ARRIVAL TIME*
*  P3   TXN COMPL TIME  *
*  P4   TXN EXEC TIME   *
*  P5   DEVICE ID       *
*  P11  DUMMY           *
*                       *
*************************

*************************
*                       *
*  MODEL COMPONENTS     *
*                       *
* BUSES: GBUS, LBUS1,...*
* CACHES: D11....D15    *
* LEVEL CONTRL: K1,K2   *
* REQ PROCS: E2         *
* DEVICES: D21          *
* STORAGE : RI, RO      *
* STORAGE : SI, SO      *
* STORAGE : TI, TO      *
* STORAGE : AI, AO      *
* STORAGE : CI, OO      *
*                       *
*************************

*************************
*                       *
* MODEL PARAMETERS      *
*                       *
*************************
```

```
      INITIAL    XS$MODEL.402       MODEL CC/PTB/C
      INITIAL    XS$MAXMP.10        DEGREE OF MULTIPROG PER CPU
      INITIAL    XS$READ.700        % READ REQ
      INITIAL    XS$PIN1.900        CONDITIONAL PROB OF FINDING DATA IN L1
      INITIAL    XS$PIN2.1000       IN L2
      INITIAL    XS$DEV1.2          DEVICE SERVICE TIME IN L1
       INITIAL    XS$DEV2.4         IN L2
      INITIAL    XS$BE1.1           BUS SERVICE TIME FOR 8 BYTE BLOCK
      INITIAL    XS$BE8.8           BUS SERVICE TIME FOR 64 BYTE BLOCK
      INITIAL    XS$DE.4            DIRECTORY LOOK UP
      INITIAL    XS$CEX.4           CONTROLLER SERV TIME
      INITIAL    XS$P_R.8           TIME TO USE REPL ALG AND STORE IN
                                       LEVEL (1)
      INITIAL    XS$DEX1.6          LOOKUP PLUS READ/WRITE TIME OF D11
      INITIAL    XSTIMER.10000      SIMULATION TIME (IN 10 NS UNITS)
      INITIAL    XS$R1S.15          TIME R1 SEARCHS DIREC & UPDT LRU STATUS

*************************
*                       *
* SAVEVALUES            *
*                       *
* NTXN TOTAL TXN PROC.  *
* SUMX TOTAL EXEC TIMES *
* SUMW TOTAL WAIT TIMES *
* SUMT TOTAL ELAPSED TIM*
*                       *
*************************

*************************
*                       *
*  VARIABLES            *
*                       *
*************************

 MRESP FVARIABLE  (XS$SUMT/XS$NTXN)        MEAN RESP TIME
 TXNT  VARIABLE   P3-P2                     TXN ELAPSED TIME
 TXNW  VARIABLE   P3-P2-P4                  TXN WAIT TIME
 TXNX  VARIABLE   P4                        TXN EXEC TIME

*************************
*                       *
*  TABLES               *
*                       *
*************************

 TXNT  TABLE      V$TXNT,100,100,100
 TXNW  TABLE      V$TXNW,100,100,100
 TXNX  TABLE      V$TXNX,100,100,100

*************************
*                       *
*  FUNCTIONS            *
*                       *
*************************
 WCHST FUNCTION   P1,D5
```

```
1,STAR1/2,STAR2/3,STAR3/4,STAR4/5,STAR5

 WICHW FUNCTION    P5,D5
1,WWW11/2,WWW12/3,WWW13/4,WWW14,5,WWW15

 WUNIT FUNCTION    P5,D5
1,WDD11/2,WDD12/3,WDD13/4,WDD14/5,WDD15

 RUNIT FUNCTION    P5,D5
1,RDD11/2,RDD12/3,RDD13/4,RDD14/5,RDD15

 LOAD  FUNCTION    RN1,D5
.2,1/.4,2/.6,3/.8,4/1,5

 WICHA FUNCTION    P5,D5
1,AAA11/2,AAA12/3,AAA13/4,AAA14/5,AAA15
*************************
*                       *
* STORAGE FOR L(1)      *
* CACHES                *
*                       *
*************************

      STORAGE    S$RID11,10/S$SID11,10/S$TID11,10/S$AID11,10
      STORAGE    S$RID12,10/S$SID12,10/S$TID12,10/S$AID12,10
      STORAGE    S$RID13,10/S$SID13,10/S$TID13,10/S$AID13,10
      STORAGE    S$RID14,10/S$SID14,10/S$TID14,10/S$AID14,10
      STORAGE    S$RID15,10/S$SID15,10/S$TID15,10/S$AID15,10

*************************
*                       *
* STORAGE FOR DEVICES   *
*                       *
*************************

      STORAGE    S$RID21,10/S$SID21,10/S$TID21,10
*************************
*                       *
* STORAGE FOR REQ PROC  *
*                       *
*************************

      STORAGE    S$RIR1,10/S$SIR1,10/S$TIR1,10/S$AIR1,10/S$OIR1,10
      STORAGE    S$RIR2,10/S$SIR2,10/S$TIR2,10/S$AIR2,10/S$OIR2,10

*************************
*                       *
*  STORAGE FOR K1       *
*                       *
*************************

      STORAGE    S$ROK1,10/S$SOK1,10/S$TIK1,10/S$AIK1,10/S$OCK1,10

*************************
*                       *
```

```
* STORAGE FOR K2,K3,K4  *
*                       *
*************************

       STORAGE   S$RIK2,10/S$SIK2,10/S$TIK2,10/S$AIK2,10/S$OIK2,10
       STORAGE   S$ROK2,10/S$SOK2,10/S$TOK2,10/S$AOK2,10/S$OOK2,10

*************************
*                       *
* BOOLEAN VARIABLES     *
*                       *
*************************

*************************
*                       *
* BV FOR READ-THROUGH   *
*                       *
*************************

RTOK2 BVARIABLE  FNU$GBUS*SNF$TIK1

*************************
*                       *
* BV FOR L(1)           *
*                       *
*************************
CRR1  BVARIABLE  FNU$LBUS1*SNF$RIR1
RDR11 BVARIABLE  FNU$LBUS1*SNF$RID11
RDR12 BVARIABLE  FNU$LBUS1*SNF$RID12
RDR13 BVARIABLE  FNU$LBUS1*SNF$RID13
RDR14 BVARIABLE  FNU$LBUS1*SNF$RID14
RDR15 BVARIABLE  FNU$LBUS1*SNF$RID15
RDS11 BVARIABLE  FNU$LBUS1*SNF$SID11
RDS12 BVARIABLE  FNU$LBUS1*SNF$SID12
RDS13 BVARIABLE  FNU$LBUS1*SNF$SID13
RDS14 BVARIABLE  FNU$LBUS1*SNF$SID14
RDS15 BVARIABLE  FNU$LBUS1*SNF$SID15
KRT1  BVARIABLE  FNU$LBUS1*SNF$TIR1
KRA1  BVARIABLE  FNU$LBUS1*SNF$AIR1

DKR1  BVARIABLE  FNU$LBUS1*SNF$ROK1
DKS1  BVARIABLE  FNU$LBUS1*SNF$SOK1
DKO1  BVARIABLE  FNU$LBUS1*SNF$OOK1
KDT11 BVARIABLE  FNU$LBUS1*SNF$TID11
KDT12 BVARIABLE  FNU$LBUS1*SNF$TID12
KDT13 BVARIABLE  FNU$LBUS1*SNF$TID13
KDT14 BVARIABLE  FNU$LBUS1*SNF$TID14
KDT15 BVARIABLE  FNU$LBUS1*SNF$TID15
KDA11 BVARIABLE  FNU$LBUS1*SNF$AID11
KDA12 BVARIABLE  FNU$LBUS1*SNF$AID12
KDA13 BVARIABLE  FNU$LBUS1*SNF$AID13
KDA14 BVARIABLE  FNU$LBUS1*SNF$AID14
KDA15 BVARIABLE  FNU$LBUS1*SNF$AID15

*************************
```

```
*                        *
* BV FOR INTER LEVEL COM*
*                        *
**************************

KKR12 BVARIABLE   FNU$GBUS*SNF$RIK2
KKS12 BVARIABLE   FNU$GBUS*SNF$SIK2
KKO12 BVARIABLE   FNU$GBUS*SNF$SOK2
KKT21 BVARIABLE   FNU$GBUS*SNF$TIK1
KKA21 BVARIABLE   FNU$GBUS*SNF$AIK1

**************************
*                        *
* BV FOR L(2) OPS        *
*                        *
**************************

KRR2  BVARIABLE   FNU$LBUS2*SNF$RIR2
KRS2  BVARIABLE   FNU$LBUS2*SNF$SIR2
KRT2  BVARIABLE   FNU$LBUS2*SNF$TIR2
KRA2  BVARIABLE   FNU$LBUS2*SNF$AIR2
KRO2  BVARIABLE   FNU$LBUS2*SNF$OIR2
RDR21 BVARIABLE   FNU$LBUS2*SNF$RID21
RDS21 BVARIABLE   FNU$LBUS2*SNF$SID21
RDT21 BVARIABLE   FNU$LBUS2*SNF$TID21
DKS2  BVARIABLE   FNU$LBUS2*SNF$SOK2
DKT2  BVARIABLE   FNU$LBUS2*SNF$TOK2
DKA2  BVARIABLE   FNU$LBUS2*SNF$AOK2
RKR2  BVARIABLE   FNU$LBUS2*SNF$ROK2
RKO2  BVARIABLE   FNU$LBUS2*SNF$OOK2
RKA2  BVARIABLE   FNU$LBUS2*SNF$AOK2

**************************
*                        *
*    MACROS              *
*                        *
**************************

**************************
*                        *
* MACRO -USE             *
* #A  FACILITY           *
*
* #B  USAGE TIME

USE   STARTMACRO
      SEIZE      #A
      ADVANCE    #B
      ASSIGN     4+,#B
      RELEASE    #A
      ENDMACRO

**************************
*                        *
```

```
*  MACRO - SEND            *
*                          *
* #A  FROM                 *
* #B  TO                   *
* #C  VIA                  *
* #D  TRANSIT TIME         *
* #E  BV FOR SEND OP      *
*                          *
****************************

 SEND  STARTMACRO
       TEST E      #E,1
       ENTER       #B
       SEIZE       #C
       ADVANCE     #D
       ASSIGN      4+,#D
       RELEASE     #C
       LEAVE       #A
       ENDMACRO

****************************
*                          *
*  MACRO - SND             *
*                          *
****************************

 SND   STARTMACRO
       TEST E      #D,1
       ENTER       #A
       SEIZE       #B
       ADVANCE     #C
       ASSIGN      4+,#C
       RELEASE     #B
       ENDMACRO
****************************
*                          *
*  MACRO - FINI            *
*                          *
****************************

 FINI  STARTMACRO
       MARK        3
       SAVEVALUE   NTXN+,1
       SAVEVALUE   SUMX+,V$TXNX
       SAVEVALUE   SUMW+,V$TXNW
       SAVEVALUE   SUMT+,V$TXNT
       SAVEVALUE   MRESP,V$MRESP
       ASSIGN      2,0
       ASSIGN      3,0
       ASSIGN      4,0
       ENDMACRO

**-------------------------------------------------------*
*                                                        *
*          BEGIN SIMULATION                              *
```

```
*                                                            *
*------------------------------------------------------------*

      SIMULATE
************************
*                      *
*   CPU #1             *
*                      *
************************

      RMULT      3,5,7,9,11,13,15,17

CPU1  GENERATE   ,,,X$MAXMP,,,F
STAR1 PRIORITY   9                SET HIGH P FOR NEW TXN
      MARK       2                ARRIVAL TIME
      ASSIGN     1,1              CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #2             *
*                      *
************************

CPU2  GENERATE   ,,,X$MAXMP,,,F
STAR2 PRIORITY   9                SET HIGH P FOR NEW TXN
      MARK       2                ARRIVAL TIME
      ASSIGN     1,2              CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #3             *
*                      *
************************

CPU3  GENERATE   ,,,X$MAXMP,,,F
STAR3 PRIORITY   9                SET HIGH P FOR NEW TXN
      MARK       2                ARRIVAL TIME
      ASSIGN     1,3              CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #4             *
*                      *
************************

CPU4  GENERATE   ,,,X$MAXMP,,,F
STAR4 PRIORITY   9                SET HIGH P FOR NEW TXN
      MARK       2                ARRIVAL TIME
      ASSIGN     1,4              CPU ID
      TRANSFER   .X$NREAD,WWW1,READ
```

```
************************
*                      *
*   CPU #5             *
*                      *
************************

 CPU5  GENERATE   ,,,XSMAXMP,,,F
 STARS PRIORITY   9                  SET HIGH P FOR NEW TXN
       MARK       2                  ARRIVAL TIME
       ASSIGN     1,5                CPU ID
       TRANSFER   .XSNREAD,WWW1,READ

****************************************
*                                      *
*  DATA TO BE READ                     *
*                                      *
****************************************


 READ  ASSIGN     11,0
       ASSIGN     5,FNSLOAD
       TRANSFER   ,FNSRUNIT

************************
*                      *
* DEVICE D11           *
*                      *
************************

 RDD11 ASSIGN     5,1
 SND   MACRO      RID11,LBUS1,X$BEX1,BV$RDR11
       TRANSFER   .X$PIN1,NIN11,RIN11
 RIN11 ASSIGN     11,0
 USE   MACRO      DRP11,X$RDEX1
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4+,X$BEX1
       RELEASE    LBUS1
       LEAVE      RID11
 FINI  MACRO
       SPLIT      1,FN$WCHST
       TERMINATE
 NIN11 ASSIGN     11,0
       PRIORITY   0
 USE   MACRO      DRP11,X$REX
 SEND  MACRO      RID11,ROK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER   ,COMR

************************
*                      *
* DEVICE D12           *
*                      *
```

```
************************

RDD12 ASSIGN     5.2
SND   MACRO      RID12,LBUS1,X$BEX1,BV$RDR12
      TRANSFER   .X$PIN1,NIN12,RIN12
RIN12 ASSIGN     11,0
USE   MACRO      DRP12,X$RDEX1
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      RID12
FINI  MACRO
      SPLIT      1,FN$WCHST
      TERMINATE
NIN12 ASSIGN     11,0
      PRIORITY   0
USE   MACRO      DRP12,X$REX
SEND  MACRO      RID12,RDK1,LBUS1,X$BEX1,BV$DKR1
      TRANSFER   ,COMR

************************
*                      *
* DEVICE D13           *
*                      *
************************

RDD13 ASSIGN     5.3
SND   MACRO      RID13,LBUS1,X$BEX1,BV$RDR13
      TRANSFER   .X$PIN1,NIN13,RIN13
RIN13 ASSIGN     11,0
USE   MACRO      DRP13,X$RDEX1
      SEIZE      LBUS1
      ADVANCE    X$BEX1
      ASSIGN     4+,X$BEX1
      RELEASE    LBUS1
      LEAVE      RID13
FINI  MACRO
      SPLIT      1,FN$WCHST
      TERMINATE
NIN13 ASSIGN     11,0
      PRIORITY   0
USE   MACRO      DRP13,X$REX
SEND  MACRO      RID13,RDK1,LBUS1,X$BEX1,BV$DKR1
      TRANSFER   ,COMR

************************
*                      *
* DEVICE D14           *
*                      *
************************

RDD14 ASSIGN     5.4
```

```
SND    MACRO       RID14,LBUS1,X$BEX1,BV$RDR14
       TRANSFER    .X$PIN1,NIN14,RIN14
RIN14  ASSIGN      11,0
USE    MACRO       DRP14,X$RDEX1
       SEIZE       LBUS1
       ADVANCE     X$BEX1
       ASSIGN      4+,X$BEX1
       RELEASE     LBUS1
       LEAVE       RID14
FINI   MACRO
       SPLIT       1,FN$WCHST
       TERMINATE
NIN14  ASSIGN      11,0
       PRIORITY    0
USE    MACRO       DRP14,X$REX
SEND   MACRO       RID14,ROK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER    ,COMR

************************
*                      *
* DEVICE D15           *
*                      *
************************

RDD15  ASSIGN      5,5
SND    MACRO       RID15,LBUS1,X$BEX1,BV$RDR15
       TRANSFER    .X$PIN1,NIN15,RIN15
RIN15  ASSIGN      11,0
USE    MACRO       DRP15,X$RDEX1
       SEIZE       LBUS1
       ADVANCE     X$BEX1
       ASSIGN      4+,X$BEX1
       RELEASE     LBUS1
       LEAVE       RID15
FINI   MACRO
       SPLIT       1,FN$WCHST
       TERMINATE
NIN15  ASSIGN      11,0
       PRIORITY    0
USE    MACRO       DRP15,X$REX
SEND   MACRO       RID15,ROK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER    ,COMR

**************************************
*                                    *
*  DATA TO BE WRITTEN IS IN LEVEL 1  *
*                                    *
**************************************

WWW1   ASSIGN      11,0
```

```
       ASSIGN     5,FN$LOAD
       TRANSFER   ,FN$WUNIT

*************************
*                       *
*  DEVICE D11           *
*                       *
*************************

WDD11  ASSIGN     5.1
SND    MACRO      SID11,LBUS1,X$BEX1,BV$RDS11
USE    MACRO      DRP11,X$RDEX1
       PRIORITY   0
SEND   MACRO      SID11,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT      1,COMW
FINI   MACRO
       SPLIT      1,FN$WCHST
       TERMINATE

*************************
*                       *
*  DEVICE D12           *
*                       *
*************************

WDD12  ASSIGN     5.2
SND    MACRO      SID12,LBUS1,X$BEX1,BV$RDS12
USE    MACRO      DRP12,X$RDEX1
       PRIORITY   0
SEND   MACRO      SID12,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT      1,COMW
FINI   MACRO
       SPLIT      1,FN$WCHST
       TERMINATE

*************************
*                       *
*  DEVICE D13           *
*                       *
*************************

WDD13  ASSIGN     5.3
SND    MACRO      SID13,LBUS1,X$BEX1,BV$RDS13
USE    MACRO      DRP13,X$RDEX1
       PRIORITY   0
SEND   MACRO      SID13,SOK1,LBUS1,X$BEX8,BV$DKS1
       SPLIT      1,COMW
FINI   MACRO
       SPLIT      1,FN$WCHST
       TERMINATE
```

```
*************************
*                       *
*  DEVICE D14           *
*                       *
*************************

WDD14 ASSIGN      5,4
SND   MACRO       SID14,LBUS1,X$BEX1,BV$RDS14
USE   MACRO       DRP14,X$RDEX1
      PRIORITY    0
SEND  MACRO       SID14,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

*************************
*                       *
*  DEVICE D15           *
*                       *
*************************

WDD15 ASSIGN      5,5
SND   MACRO       SID15,LBUS1,X$BEX1,BV$RDS15
USE   MACRO       DRP15,X$RDEX1
      PRIORITY    0
SEND  MACRO       SID15,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

*-----------------------------------------------------*
*                                                     *
*  COMMON CODE FOR READ REQUEST                       *
*                                                     *
*-----------------------------------------------------*

COMR  ASSIGN      11,0

USE   MACRO       KRP1,X$KEX

SEND  MACRO       ROK1,RIK2,GBUS,X$BEX1,BV$KKR12

USE   MACRO       KRP2,X$KEX

SEND  MACRO       RIK2,RIR2,LBUS2,X$BEX1,BV$KRR2

USE   MACRO       RRP2,X$REX
```

```
*-------------------------------------------------------*
*                                                       *
*  READ DATA IS FOUND IN L(2)                           *
*                                                       *
*-------------------------------------------------------*

*************************
*                       *
* DATA IS IN D21        *
*                       *
*************************

RRR21 ASSIGN     11,0
SEND  MACRO      RIR2,RID21,LBUS2,X$BEX1,BV$RDR21
USE   MACRO      DRP21,X$DEX2
SEND  MACRO      RID21,TOK2,LBUS2,X$BEX8,BV$DKT2

*************************
*                       *
* READ-THROUGH TO L(1)  *
*                       *
*************************

USE   MACRO      KRP2,X$KEX
SEND  MACRO      TOK2,TIK1,GBUS,X$BEX8,BV$RTOK2
*-------------------------------------------------------*
*                                                       *
* STORE DATA INTO L(1) AS RESULT OF A READ-THROUGH      *
*                                                       *
*-------------------------------------------------------*

STOR1 ASSIGN     11,0
USE   MACRO      KRP1,X$KEX

      SPLIT      1,FN$WICHW
      TERMINATE

*************************
*                       *
* RT STORE INTO D11     *
*                       *
*************************
```

```
WWW11 ASSIGN      11,0

SEND  MACRO       TIK1,TID11,LBUS1,X$BEX8,BV$KDT11

USE   MACRO       DRP11,X$RPLR

SEND  MACRO       TID11,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT       1,OVF11
FINI  MACRO
      SPLIT       1,FNS$CHST
      TERMINATE

OVF11 TRANSFER    ,OVL1

*************************
*                       *
* RT STORE INTO D12     *
*                       *
*************************

WWW12 ASSIGN      11,0

SEND  MACRO       TIK1,TID12,LBUS1,X$BEX8,BV$KDT12

USE   MACRO       DRP12,X$RPLR

SEND  MACRO       TID12,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT       1,OVF12
FINI  MACRO
      SPLIT       1,FNSWCHST
      TERMINATE

OVF12 TRANSFER    ,OVL1

*************************
*                       *
* RT STORE INTO D13     *
*                       *
*************************

WWW13 ASSIGN      11,0

SEND  MACRO       TIK1,TID13,LBUS1,X$BEX8,BV$KDT13

USE   MACRO       DRP13,X$RPLR

SEND  MACRO       TID13,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT       1,OVF13
FINI  MACRO
      SPLIT       1,FNSWCHST
      TERMINATE
```

```
OVF13 TRANSFER   ,OVL1

************************
*                      *
* RT STORE INTO D14    *
*                      *
************************

WWW14 ASSIGN      11,0

SEND  MACRO       TIK1,TID14,LBUS1,X$BEX8,BV$KDT14

USE   MACRO       DRP14,X$RPLR

SEND  MACRO       TID14,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT       1,OVF14
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

OVF14 TRANSFER   ,OVL1

************************
*                      *
* RT STORE INTO D15    *
*                      *
************************

WWW15 ASSIGN      11,0

SEND  MACRO       TIK1,TID15,LBUS1,X$BEX8,BV$KDT15

USE   MACRO       DRP15,X$RPLR

SEND  MACRO       TID15,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT       1,OVF15
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

OVF15 TRANSFER   ,OVL1

************************
*                      *
* HANDLE OVF FROM L(1) *
*                      *
************************

OVL1  ASSIGN      11,0

USE   MACRO       KRP1,X$KEX
```

```
SEND  MACRO     DCK1,DIK2,GBUS,X$BEX1,BV$KKD12

USE   MACRO     KRP2,X$KEX

SEND  MACRO     DIK2,DIR2,LBUS2,X$BEX1,BV$KRD2

USE   MACRO     RRP2,X$REX

      LEAVE     DIR2
      TERMINATE

*--------------------------------------------------------*
*                                                        *
*  COMMON CODE FOR STORE-BEHIND                          *
*                                                        *
*--------------------------------------------------------*

COMM  ASSIGN    11,C

USE   MACRO     KRP1,X$KEX

SEND  MACRO     SCK1,SIK2,GBUS,X$BEX8,BV$KKS12

USE   MACRO     KRP2,X$KEX

SEND  MACRO     SIK2,SIR2,LBUS2,X$BEX8,BV$KRS2

USE   MACRO     RRP2,X$REX

SEND  MACRO     SIR2,SID21,LBUS2,X$BEX8,BV$RDS21

USE   MACRO     DRP21,X$DEX2

SEND  MACRO     SID21,ADK2,LBUS2,X$BEX1,BV$DKS2

*--------------------------------------------------------*
*                                                        *
*  ACK FROM L(2) TO L(1)                                 *
*                                                        *
*--------------------------------------------------------*

USE   MACRO     KRP2,X$KEX

SEND  MACRO     ACK2,AIK1,GBUS,X$BEX1,BV$KKA21

USE   MACRO     KRP1,X$KEX

      SPLIT     1,FN$WICHA
      TERMINATE
```

```
************************
*                      *
* ACK HANDLED BY D11   *
*                      *
************************

AAA11 ASSIGN     11,0

SEND  MACRO      AIK1,AID11,LBUS1,X$BEX1,BV$KDA11

USE   MACRO      DRP11,X$REX

      LEAVE      AID11
      TERMINATE

************************
*                      *
* ACK HANDLED BY D12   *
*                      *
************************

AAA12 ASSIGN     11,0

SEND  MACRO      AIK1,AID12,LBUS1,X$BEX1,BV$KDA12

USE   MACRO      DRP12,X$REX

      LEAVE      AID12
      TERMINATE

************************
*                      *
* ACK HANDLED BY D13   *
*                      *
************************

AAA13 ASSIGN     11,0

SEND  MACRO      AIK1,AID13,LBUS1,X$BEX1,BV$KDA13

USE   MACRO      DRP13,X$REX

      LEAVE      AID13
      TERMINATE

************************
*                      *
* ACK HANDLED BY D14   *
*                      *
************************

AAA14 ASSIGN     11,0

SEND  MACRO      AIK1,AID14,LBUS1,X$BEX1,BV$KDA14
```

```
USE   MACRO        DRP14,XSREX

      LEAVE        AID14
      TERMINATE

**************************
*                        *
* ACK HANDLED BY D15     *
*                        *
**************************

AAA15 ASSIGN       11,0

SEND  MACRO        AIK1,AID15,LBUS1,X$BEX1,BV$KDA15

USE   MACRO        DRP15,X$REX

      LEAVE        AID15
      TERMINATE

*--------------------------------------------------------*
*                                                        *
*  SIMULATION CONTROL                                    *
*                                                        *
*--------------------------------------------------------*

      GENERATE     X$TIMER
      TERMINATE    1
      START        1
      END
$$ END
```

APPENDIX (IV): The "CC/PTB/P" Program

```
//TAR1 JOB TAR,
// PROFILE='DEFER',
// TIME=5
//*PASSWORD SCJBA
//GPSS PROC
//C    EXEC   PGM=DAG01,TIME=&TLIMIT
//STEPLIB DD DSN=POTLUCK.LIBRARY.GPSS.LOAD,DISP=SHR
//DOUTPUT DD SYSOUT=PROFILE=PRINT,DCB=BLKSIZE=931
//DINTEPO DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=1880
//DSYMTAB DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=7112
//DREPTGEN DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=800
//DINTWORK DD UNIT=SCRATCH,SPACE=(CYL,(1,1)),DCB=BLKSIZE=2680
// PEND
//STEP1  EXEC  GPSS,PARM=C,TLIMIT=9
//DINPUT1  DD *
        REALLOCATE FUN,6,QUE,10,FAC,50,BVR,200,BLO,2000,VAR,50
        REALLOCATE FSV,50,HSV,10,COM,39968

******************************  CC/PTB/P
*************************
*                       *
* TXN PARM USAGE        *
*                       *
*  P1   CPU ID          *
*  P2   TXN ARRIVAL TIME*
*  P3   TXN COMPL TIME  *
*  P4   TXN EXEC TIME   *
*  P5   DEVICE ID       *
*  P11  DUMMY           *
*                       *
*************************

*************************
*                       *
*  MODEL COMPONENTS     *
*                       *
* BUSES: GBUS, LBUS1,...*
* CACHES: D11....D15    *
* LEVEL CONTRL: K1,K2   *
* REQ PROCS: R2         *
* DEVICES: D21          *
* STORAGE : RI, RO      *
* STORAGE : SI, SO      *
* STORAGE : TI, TO      *
* STORAGE : AI, AO      *
* STORAGE : OI, OO      *
*                       *
*************************

*************************
*                       *
* MODEL PARAMETERS      *
*                       *
*************************
```

```
       INITIAL    X$MODEL,502     MODEL CC/PTB/P
       INITIAL    X$MAXMP,10      DEGREE OF MULTIPROG PER CPU
       INITIAL    X$NREAD,700     % READ REQ
       INITIAL    X$PIN1,900      CONDITIONAL PROB OF FINDING DATA IN L1
       INITIAL    X$PIN2,1000     IN LL2
       INITIAL    X$DEX1,2        DEVICE SERVICE TIME IN L1
       INITIAL    X$DEX2,4        IN L2
       INITIAL    X$BEX1,1        BUS SERVICE TIME FOR 8 BYTE BLOCK
       INITIAL    X$BEX8,8        BUS SERVICE TIME FOR 64 BYTE BLOCK
       INITIAL    X$REX,4         DIRECTORY LOOK UP TIME
       INITIAL    X$KEX,4         CONTROLLER SERV TIME
       INITIAL    X$RPLR,8        TIME TO USE REPL ALG AND STORE IN
*                                    LEVEL (1)
       INITIAL    X$RDEX1,6       LOOKUP PLUS READ/WRITE TIME OF D11
       INITIAL    X$TIMER,10000   SIMULATION TIME (IN 10 NS UNITS)
       INITIAL    X$R1SU,15       TIME R1 SEARCHS DIREC & UPDT LRU STATUS

*************************
*                       *
* SAVEVALUES            *
*                       *
* NTXN TOTAL TXN PROC.  *
* SUMX TOTAL EXEC TIMES *
* SUMW TOTAL WAIT TIMES *
* SUMT TOTAL ELAPSED TIM*
*                       *
*************************

*************************
*                       *
*  VARIABLES            *
*                       *
*************************

 MRESP FVARIABLE  (X$SUMT/X$NTXN)        MEAN RESP TIME
 TXNT  VARIABLE   P3-P2                  TXN ELAPSED TIME
 TXNW  VARIABLE   P3-P2-P4               TXN WAIT TIME
 TXNX  VARIABLE   P4                     TXN EXEC TIME

*************************
*                       *
*  TABLES              *
*                       *
*************************

 TXNT  TABLE      V$TXNT,100,100,100
 TXNW  TABLE      V$TXNW,100,100,100
 TXNX  TABLE      V$TXNX,100,100,100

*************************
*                       *
*  FUNCTIONS            *
*                       *
*************************
 WCHST FUNCTION   P1,D5
```

```
1,STAR1/2,STAR2/3,STAR3/4,STAR4/5,STAR5

 WICHW FUNCTION    P5.D5
1,WWW11/2,WWW12/3,WWW13/4,WWW14/5,WWW15

 WUNIT FUNCTION    P5.D5
1,WDD11/2,WDD12/3,WDD13/4,WDD14/5,WDD15

 RUNIT FUNCTION    P5.D5
1,RDD11/2,RDD12/3,RDD13/4,RDD14/5,RDD15

 LOAD  FUNCTION    RN1,D5
.2,1/.4,2/.6,3/.8,4/1,5

************************
*                      *
* STORAGE FOR L(1)     *
* CACHES               *
*                      *
************************

       STORAGE    S$RID11,10/S$SID11,10/S$TID11,10/S$AID11,10
       STORAGE    S$RID12,10/S$SID12,10/S$TID12,10/S$AID12,10
       STORAGE    S$RID13,10/S$SID13,10/S$TID13,10/S$AID13,10
       STORAGE    S$RID14,10/S$SID14,10/S$TID14,10/S$AID14,10
       STORAGE    S$RID15,10/S$SID15,10/S$TID15,10/S$AID15,10

************************
*                      *
* STORAGE FOR DEVICES  *
*                      *
************************

       STORAGE    S$RID21,10/S$SID21,10/S$TID21,10
************************
*                      *
* STORAGE FOR REQ PROC *
*                      *
************************

       STORAGE    S$RIR1,10/S$SIR1,10/S$TIR1,10/S$AIR1,10/S$DIR1,10
       STORAGE    S$RIR2,10/S$SIR2,10/S$TIR2,10/S$AIR2,10/S$DIR2,10

************************
*                      *
*  STORAGE FOR K1      *
*                      *
************************

       STORAGE    S$RCK1,10/S$SCK1,10/S$TIK1,10/S$AIK1,10/S$DOK1,10

************************
*                      *
* STORAGE FOR K2,K3,K4 *
*                      *
```

```
*************************

       STORAGE    S$RIK2,10/S$SIK2,10/S$TIK2,10/S$AIK2,10/S$OIK2,10
       STORAGE    S$ROK2,10/S$SOK2,10/S$TOK2,10/S$AOK2,10/S$OOK2,10

*************************
*                       *
* BOOLEAN VARIABLES     *
*                       *
*************************

*************************
*                       *
* BV FOR READ-THROUGH   *
*                       *
*************************

 RTOK2 BVARIABLE  FNU$GBUS*SNF$TIK1

*************************
*                       *
* BV FOR L(1)           *
*                       *
*************************
 CRR1  BVARIABLE  FNU$LBUS1*SNF$RIR1
 RDR11 BVARIABLE  FNU$LBUS1*SNF$RID11
 RDR12 BVARIABLE  FNU$LBUS1*SNF$RID12
 RDR13 BVARIABLE  FNU$LBUS1*SNF$RID13
 RDR14 BVARIABLE  FNU$LBUS1*SNF$RID14
 RDR15 BVARIABLE  FNU$LBUS1*SNF$RID15
 RDS11 BVARIABLE  FNU$LBUS1*SNF$SID11
 RDS12 BVARIABLE  FNU$LBUS1*SNF$SID12
 RDS13 BVARIABLE  FNU$LBUS1*SNF$SID13
 RDS14 BVARIABLE  FNU$LBUS1*SNF$SID14
 RDS15 BVARIABLE  FNU$LBUS1*SNF$SID15
 KRT1  BVARIABLE  FNU$LBUS1*SNF$TIR1
 KRA1  BVARIABLE  FNU$LBUS1*SNF$AIR1

 DKR1  BVARIABLE  FNU$LBUS1*SNF$ROK1
 DKS1  BVARIABLE  FNU$LBUS1*SNF$SOK1
 DKO1  BVARIABLE  FNU$LBUS1*SNF$OOK1
 KDT11 BVARIABLE  FNU$LBUS1*SNF$TID11
 KDT12 BVARIABLE  FNU$LBUS1*SNF$TID12
 KDT13 BVARIABLE  FNU$LBUS1*SNF$TID13
 KDT14 BVARIABLE  FNU$LBUS1*SNF$TID14
 KDT15 BVARIABLE  FNU$LBUS1*SNF$TID15
 KDA11 BVARIABLE  FNU$LBUS1*SNF$AID11
 KDA12 BVARIABLE  FNU$LBUS1*SNF$AID12
 KDA13 BVARIABLE  FNU$LBUS1*SNF$AID13
 KDA14 BVARIABLE  FNU$LBUS1*SNF$AID14
 KDA15 BVARIABLE  FNU$LBUS1*SNF$AID15

*************************
*                       *
* BV FOR INTER LEVEL COM*
```

```
*                       *
*************************

 KKR12 BVARIABLE   FNUSGBUS*SNFSRIK2
 KKS12 BVARIABLE   FNUSGBUS*SNFSSIK2
 KKO12 BVARIABLE   FNUSGBUS*SNFSSOK2
 KKT21 BVARIABLE   FNUSGBUS*SNFSTIK1
 KKA21 BVARIABLE   FNUSGBUS*SNFSAIK1

*************************
*                       *
* BV FOR L(2) CPS       *
*                       *
*************************

 KRR2  BVARIABLE   FNUSLBUS2*SNFSRIR2
 KRS2  BVARIABLE   FNUSLBUS2*SNFSSIR2
 KRT2  BVARIABLE   FNUSLBUS2*SNFSTIR2
 KRA2  BVARIABLE   FNUSLBUS2*SNFSAIR2
 KRO2  BVARIABLE   FNUSLBUS2*SNFSOIR2
 RDR21 BVARIABLE   FNUSLBUS2*SNFSRID21
 RDS21 BVARIABLE   FNUSLBUS2*SNFSSID21
 RDT21 BVARIABLE   FNUSLBUS2*SNFSTID21
 DKS2  BVARIABLE   FNUSLBUS2*SNFSSOK2
 DKT2  BVARIABLE   FNUSLBUS2*SNFSTOK2
 DKA2  BVARIABLE   FNUSLBUS2*SNFSAOK2
 RKR2  BVARIABLE   FNUSLBUS2*SNFSROK2
 RKO2  BVARIABLE   FNUSLBUS2*SNFSOOK2
 RKA2  BVARIABLE   FNUSLBUS2*SNFSAOK2

*************************
*                       *
*    MACROS             *
*                       *
*************************

*************************
*                       *
* MACRO -USE            *
* #A  FACILITY         *
*
* #B  USAGE TIME

 USE   STARTMACRO
       SEIZE      #A
       ADVANCE    #B
       ASSIGN     4+,#B
       RELEASE    #A
       ENDMACRO

*************************
*                       *
*  MACRO - SEND         *
*                       *
```

```
* #A  FROM              *
* #B  TO                *
* #C  VIA               *
* #D  TRANSIT TIME      *
* #E  BV FOR SEND OP    *
*                        *
**************************

 SEND  STARTMACRO
       TEST E      #E,1
       ENTER       #B
       SEIZE       #C
       ADVANCE     #D
       ASSIGN      4+,#D
       RELEASE     #C
       LEAVE       #A
       ENDMACRO

**************************
*                        *
*  MACRO - SND           *
*                        *
**************************

 SND   STARTMACRO
       TEST E      #D,1
       ENTER       #A
       SEIZE       #B
       ADVANCE     #C
       ASSIGN      4+,#C
       RELEASE     #B
       ENDMACRO
**************************
*                        *
*  MACRO - FINI          *
*                        *
**************************

 FINI  STARTMACRO
       MARK        3
       SAVEVALUE   NTXN+,1
       SAVEVALUE   SUMX+,VSTXNX
       SAVEVALUE   SUMW+,VSTXNW
       SAVEVALUE   SUMT+,VSTXNT
       SAVEVALUE   MRESP,VSMRESP
       ASSIGN      2,0
       ASSIGN      3,0
       ASSIGN      4,0
       ENDMACRO

*--------------------------------------------------------*
*                                                         *
*          BEGIN SIMULATION                               *
*                                                         *
*--------------------------------------------------------*
```

```
      SIMULATE
************************
*                      *
*   CPU #1             *
*                      *
************************

      RMULT      3,5,7,9,11,13,15,17

CPU1  GENERATE   ,,,X$MAXMP,,,F
STAR1 PRIORITY   9                 SET HIGH P FOR NEW TXN
      MARK       2                 ARRIVAL TIME
      ASSIGN     1,1               CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #2             *
*                      *
************************

CPU2  GENERATE   ,,,X$MAXMP,,,F
STAR2 PRIORITY   9                 SET HIGH P FOR NEW TXN
      MARK       2                 ARRIVAL TIME
      ASSIGN     1,2               CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #3             *
*                      *
************************

CPU3  GENERATE   ,,,X$MAXMP,,,F
STAR3 PRIORITY   9                 SET HIGH P FOR NEW TXN
      MARK       2                 ARRIVAL TIME
      ASSIGN     1,3               CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
*   CPU #4             *
*                      *
************************

CPU4  GENERATE   ,,,X$MAXMP,,,F
STAR4 PRIORITY   9                 SET HIGH P FOR NEW TXN
      MARK       2                 ARRIVAL TIME
      ASSIGN     1,4               CPU ID
      TRANSFER   .X$NREAD,WWW1,READ

************************
*                      *
```

```
*   CPU #5               *
*                        *
**************************

 CPU5  GENERATE    ,,,X$MAXMP,,,F
 STAR5 PRIORITY    9                 SET HIGH P FOR NEW TXN
       MARK        2                 ARRIVAL TIME
       ASSIGN      1,5               CPU ID
       TRANSFER    .X$NREAD,WWW1,READ

*****************************************
*                                       *
*  DATA TO BE READ                      *
*                                       *
*****************************************


 READ  ASSIGN      11,0
       ASSIGN      5,FN$LOAD
       TRANSFER    ,FN$RUNIT

**************************
*                        *
* DEVICE D11             *
*                        *
**************************

 RDD11 ASSIGN      5,1
       TRANSFER    .X$PIN1,NIN11,RIN11
 RIN11 ASSIGN      11,0
 SND   MACRO       RID11,LBUS1,X$BEX1,BV$RDR11
 USE   MACRO       DRP11,X$DEX1
       SEIZE       LBUS1
       ADVANCE     X$BEX1
       ASSIGN      4+,X$BEX1
       RELEASE     LBUS1
       LEAVE       RID11
 FINI  MACRO
       SPLIT       1,FN$WCHST
       TERMINATE
 NIN11 ASSIGN      11,0
       PRIORITY    0
 SND   MACRO       ROK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER    ,COMR

**************************
*                        *
* DEVICE D12             *
*                        *
**************************

 RDD12 ASSIGN      5,2
```

```
       TRANSFER   .X$PIN1,NIN12,RIN12
 RIN12 ASSIGN     11,0
 SND   MACRO      RID12,LBUS1,X$BEX1,BV$RDR12
 USE   MACRO      DRP12,X$DEX1
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4-,X$BEX1
       RELEASE    LBUS1
       LEAVE      RID12
 FINI  MACRO
       SPLIT      1,FN$#CHST
       TERMINATE
 NIN12 ASSIGN     11,0
       PRIORITY   0
 SND   MACRO      RCK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER   ,CCMR

*************************
*                       *
* DEVICE D13            *
*                       *
*************************

 RDD13 ASSIGN     5,3
       TRANSFER   .X$PIN1,NIN13,RIN13
 RIN13 ASSIGN     11,0
 SND   MACRO      RID13,LBUS1,X$BEX1,BV$RDR13
 USE   MACRO      DRP13,X$DEX1
       SEIZE      LBUS1
       ADVANCE    X$BEX1
       ASSIGN     4-,X$BEX1
       RELEASE    LBUS1
       LEAVE      RID13
 FINI  MACRO
       SPLIT      1,FN$#CHST
       TERMINATE
 NIN13 ASSIGN     11,0
       PRIORITY   0
 SND   MACRO      RCK1,LBUS1,X$BEX1,BV$DKR1
       TRANSFER   ,CCMR

*************************
*                       *
* DEVICE D14            *
*                       *
*************************

 RDD14 ASSIGN     5,4
       TRANSFER   .X$PIN1,NIN14,RIN14
 RIN14 ASSIGN     11,0
 SND   MACRO      RID14,LBUS1,X$BEX1,BV$RDR14
 USE   MACRO      DRP14,X$DEX1
       SEIZE      LBUS1
```

```
        ADVANCE    X$BEX1
        ASSIGN     4+,X$BEX1
        RELEASE    LBUS1
        LEAVE      RID14
FINI    MACRO
        SPLIT      1,FNS#CHST
        TERMINATE
NIN14   ASSIGN     11,0
        PRIORITY   0
SND     MACRO      ROK1,LBUS1,X$BEX1,BV$DKR1
        TRANSFER   ,COMR

*************************
*                       *
* DEVICE D15            *
*                       *
*************************

RDD15   ASSIGN     5,5
        TRANSFER   .X$PIN1,NIN15,RIN15
RIN15   ASSIGN     11,0
SND     MACRO      RID15,LDUS1,X$BEX1,BV$RDR15
USE     MACRO      DRP15,X$DEX1
        SEIZE      LBUS1
        ADVANCE    X$BEX1
        ASSIGN     4+,X$BEX1
        RELEASE    LBUS1
        LEAVE      RID15
FINI    MACRO
        SPLIT      1,FNS#CHST
        TERMINATE
NIN15   ASSIGN     11,0
        PRIORITY   0
SND     MACRO      ROK1,LBUS1,X$BEX1,BV$DKR1
        TRANSFER   ,COMR

****************************************
*                                      *
*  DATA TO BE WRITTEN IS IN LEVEL 1    *
*                                      *
****************************************

WWW1    ASSIGN     11,0

        ASSIGN     5,FN$LOAD
        TRANSFER   ,FN$WUNIT

*************************
*                       *
*  DEVICE D11           *
*                       *
```

```
**************************

WDD11 ASSIGN      5.1
SND   MACRO       SID11,LBUS1,X$BEX1,BV$RDS11
USE   MACRO       DRP11,X$DEX1
      PRIORITY    0
SEND  MACRO       SID11,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

**************************
*                        *
*  DEVICE D12            *
*                        *
**************************

WDD12 ASSIGN      5.2
SND   MACRO       SID12,LBUS1,X$BEX1,BV$RDS12
USE   MACRO       DRP12,X$DEX1
      PRIORITY    0
SEND  MACRO       SID12,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

**************************
*                        *
*  DEVICE D13            *
*                        *
**************************

WDD13 ASSIGN      5.3
SND   MACRO       SID13,LBUS1,X$BEX1,BV$RDS13
USE   MACRO       DRP13,X$DEX1
      PRIORITY    0
SEND  MACRO       SID13,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

**************************
*                        *
*  DEVICE D14            *
*                        *
**************************
```

```
WDD14 ASSIGN      5.4
SND   MACRO       SID14,LBUS1,X$BEX1,BV$RDS14
USE   MACRO       DRP14,X$DEX1
      PRIORITY    0
SEND  MACRO       SID14,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

*************************
*                       *
*  DEVICE D15           *
*                       *
*************************

WDD15 ASSIGN      5.5
SND   MACRO       SID15,LBUS1,X$BEX1,BV$RDS15
USE   MACRO       DRP15,X$DEX1
      PRIORITY    0
SEND  MACRO       SID15,SOK1,LBUS1,X$BEX8,BV$DKS1
      SPLIT       1,COMW
FINI  MACRO
      SPLIT       1,FN$WCHST
      TERMINATE

*-------------------------------------------------*
*                                                 *
*  COMMON CODE FOR READ REQUEST                   *
*                                                 *
*-------------------------------------------------*

COMR  ASSIGN      11,0

USE   MACRO       KRP1,X$KEX

SEND  MACRO       ROK1,RIK2,GBUS,X$BEX1,BV$KKR12

USE   MACRO       KRP2,X$KEX

SEND  MACRO       RIK2,RIR2,LBUS2,X$BEX1,BV$KRR2

USE   MACRO       RRP2,X$REX

*-------------------------------------------------*
*                                                 *
*  READ DATA IS FOUND IN L(2)                     *
*                                                 *
*-------------------------------------------------*
```

```
***************************
*                         *
* DATA IS IN D21          *
*                         *
***************************

RRR21 ASSIGN     11,0
SEND  MACRO      RDR2,RID21,LBUS2,X$BEX1,BV$RDR21
USE   MACRO      DRR21,X$DEX2
SEND  MACRO      RID21,TOK2,LBUS2,X$BEX8,BV$DKT2

***************************
*                         *
* READ-THROUGH TO D11     *
*                         *
***************************

USE   MACRO      KRR2,X$KEX
SEND  MACRO      TOK2,TIK1,GBUS,X$BEX8,BV$RTOK2
*---------------------------------------------------------*
*                                                         *
* STORE DATA INTO D11 AS RESULT OF A READ-THROUGH         *
*                                                         *
*---------------------------------------------------------*

STOR1 ASSIGN     11,0
USE   MACRO      KRR1,X$KEX

      SPLIT      1,FN$WICHM
      TERMINATE
***************************
*                         *
* RT STORE INTO D11       *
*                         *
***************************

WWW11 ASSIGN     11,0
SEND  MACRO      TIK1,TID11,LBUS1,X$BEX8,BV$KDT11
USE   MACRO      DRR11,X$DEX1
```

```
SEND  MACRO       TID11,OOK1,LBUS1,XSBEX1,BVSDKO1
      SPLIT       1,OVF11
FINI  MACRO
      SPLIT       1,FNSWCHST
      TERMINATE

OVF11 TRANSFER    ,OVL1

************************
*                      *
* RT STORE INTO D12    *
*                      *
************************

WWW12 ASSIGN      11,0

SEND  MACRO       TIK1,TID12,LBUS1,XSBEX8,BVSKDT12

USE   MACRO       DRP12,XSDEX1

SEND  MACRO       TID12,OOK1,LBUS1,XSBEX1,BVSDKO1
      SPLIT       1,OVF12
FINI  MACRO
      SPLIT       1,FNSWCHST
      TERMINATE

OVF12 TRANSFER    ,OVL1

************************
*                      *
* RT STORE INTO D13    *
*                      *
************************

WWW13 ASSIGN      11,0

SEND  MACRO       TIK1,TID13,LBUS1,XSBEX8,BVSKDT13

USE   MACRO       DRP13,XSDEX1

SEND  MACRO       TID13,OOK1,LBUS1,XSBEX1,BVSDKO1
      SPLIT       1,OVF13
FINI  MACRO
      SPLIT       1,FNSWCHST
      TERMINATE

OVF13 TRANSFER    ,OVL1

************************
*                      *
* RT STORE INTO D14    *
*                      *
```

```
*************************

WWW14 ASSIGN     11,0

SEND  MACRO      TIK1,TID14,LBUS1,X$BEX8,BV$KDT14

USE   MACRO      DRP14,X$DEX1

SEND  MACRO      TID14,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT      1,OVF14
FINI  MACRO
      SPLIT      1,FNSWCHST
      TERMINATE

OVF14 TRANSFER   ,OVL1

*************************
*                       *
* RT STORE INTO D15     *
*                       *
*************************

WWW15 ASSIGN     11,0

SEND  MACRO      TIK1,TID15,LBUS1,X$BEX8,BV$KDT15

USE   MACRO      DRP15,X$DEX1

SEND  MACRO      TID15,OOK1,LBUS1,X$BEX1,BV$DKO1
      SPLIT      1,OVF15
FINI  MACRO
      SPLIT      1,FNSWCHST
      TERMINATE

OVF15 TRANSFER   ,OVL1

*************************
*                       *
* HANDLE O/F FROM L(1)  *
*                       *
*************************

OVL1  ASSIGN     11,0

USE   MACRO      KRP1,X$KEX

SEND  MACRO      OOK1,OIK2,GBUS,X$BEX1,BV$KKO12

USE   MACRO      KRP2,X$KEX

SEND  MACRO      OIK2,OIR2,LBUS2,X$BEX1,BV$KRO2

USE   MACRO      RRP2,X$REX
```

```
       LEAVE        OIR2
       TERMINATE

*----------------------------------------------------------*
*                                                          *
*  COMMON CODE FOR STORE-BEHIND                            *
*                                                          *
*----------------------------------------------------------*

COMM  ASSIGN      11,0

USE   MACRO       KRP1,X$KEX

SEND  MACRO       SOK1,SIK2,GBUS,X$BEX8,BV$KKS12

USE   MACRO       KRP2,X$KEX

SEND  MACRO       SIK2,SIR2,LBUS2,X$BEX8,BV$KRS2

USE   MACRO       RRP2,X$REX

SEND  MACRO       SIR2,SID21,LBUS2,X$BEX8,BV$RDS21

USE   MACRO       DRP21,X$DEX2

SEND  MACRO       SID21,AOK2,LBUS2,X$BEX1,BV$DKS2

*----------------------------------------------------------*
*                                                          *
*  ACK FROM L(2) TO L(1)                                   *
*                                                          *
*----------------------------------------------------------*

USE   MACRO       KRP2,X$KEX

SEND  MACRO       AOK2,AIK1,GBUS,X$BEX1,BV$KKA21

USE   MACRO       KRP1,X$KEX
      LEAVE       AIK1

      TERMINATE

*----------------------------------------------------------*
*                                                          *
*  SIMULATION CONTROL                                      *
*                                                          *
*----------------------------------------------------------*
```

```
       GENERATE   ASTIMER
       TERMINATE  1
       START      1
       END
$$ END
```